Rigid Families for CCS and the π-calculus

Ioana Cristescu, Jean Krivine, Daniele Varacca

Rigid Families for CCS and the π-calculus

Ioana Domnina Cristescu*, Jean Krivine, and Daniele Varacca

[1] Laboratoire PPS - Université Paris Diderot
{ioana.cristescu,jean.krivine}@pps.univ-paris-diderot.fr
[2] LACL - Université Paris Est - Créteil
daniele.varacca@u-pec.fr

Abstract. This paper presents a novel causal semantics for concurrency, based on rigid families. Instead of having causality as primitive notion, in our model causality and concurrency are derived from precedence, a partial order local to each run of a process. We show that our causal semantics can interpret CCS and π-calculus terms. We propose some criteria to evaluate the correctness of a causal semantics of process calculi and we argue that none of the previous models for the π-calculus satisfy them all.

1 Introduction

Formal models for concurrency can be divided into *interleaving* models, such as *traces* and *labelled transition systems*, in which concurrency is represented as non deterministic executions, and *non interleaving* ones, like configuration structures [23], event structures [22] or presheaves [8]. Non interleaving semantics have a primitive notion of *concurrency* between computation events. As a consequence one can also derive a *causality* relation, generally defined as the complement of concurrency. These models are therefore sometimes called *causal semantics* or, if causality is represented as a partial order on events, *partial order* semantics. Causal models are also known to be at the foundations of reversible concurrent computations [19].

In this paper we propose to take a notion of *precedence* as the fundamental relation between events. Precedence is a partial order that can be seen as a temporal observation, specific to a given run of a process. In a given run, two events may also not be related by any precedence relation, in which case one can see them as having occurred either simultaneously or in such a way that no common clock can be used to compare them. More traditional causality and concurrency relations are derivable from precedence.

The interpretation of a process is built by induction on the process constructors and defined as operations on *rigid families* [15,6]. We equip rigid families with a *labelling* function on events. In this sense, our semantics resembles the encoding of CCS in configuration structures [22].

The operations on rigid families that are used to encode CCS can be easily adapted to the π-calculus. Importantly, the restriction operator behaves similarly

* Partially supported by the ANR grant (REVER) ANR-11-INSE-0007

in the rigid families for both calculi: it removes from the model the executions that are not allowed. In previous models for the π-calculus [9,2] the restriction of a private name introduced new orders between events.

Several causal models have been proposed in the literature for process calculi (see for instance Refs.[3,22,9,2,4,5,7,12,13]). Each model is shown to be correct, in some sense. But can a model be *more correct* than another one? What are the criteria one uses to make such evaluation? We will show that our model satisfies several correctness criteria, and we will argue that no previous causal model for the π-calculus satifies them all.

The correctness criteria will be introduced as we go along the formal development.

Outline. In section 2 we introduce the category of rigid families and rigid morphisms. In section 3 and section 4 we show how to interpret CCS and π-calculus, respectively, such that the models are compositional and sound. In section 5 we present the three remaining correction criteria and conclude with section 6.

2 A category of rigid families

In this section we present *rigid families*, a model for concurrency introduced by Hayman and Winskel [15,6], that is a close relative to *configuration structures* [14]. We first introduce the unlabelled categorical setting. It results in a generic framework to represent concurrent computations as sets of events (configurations) equipped with a partial order that represent temporal precedence between events. Importantly precedence is local to a configuration whereas events can occur in multiple configurations.

When a process P is able to output two messages a and b on two parallel channels, three kinds of observations are possible. The observer sees the output on a before the output on b, or the output on b before the output on a, or she cannot tell in which order they happen, either because they really happen at the same time, or because the observer does not have a global clock on the two events. In the rigid family interpretation of P we would have three corresponding configurations for the parallel emission of a and b.

Definition 1 (Rigid inclusion of partial orders). *Given a partial order x, we write $|x|$ to denote the underlying set and $e \leq_x e'$ whenever $(e, e') \in x$. Rigid inclusion of partial orders $x \preceq y$ is defined iff the following hold:*

$$|x| \subseteq |y| \text{ and } \forall e, e' \in x : e \leq_x e' \iff e \leq_y e'$$
$$\forall e \in y, \forall e' \in x, e \leq_y e' \implies e \in x$$

Definition 2 (Rigid families). *A* rigid family *$\mathcal{F} = (E, C)$ is a set of events E and a non empty family of partial orders, called* configurations*, such that $\forall x \in C$, $|x| \in \mathcal{P}(E)$ and C is downward closed w.r.t rigid inclusion: $\forall y \preceq x, y \in C$.*

A morphism on rigid families $\sigma : (E, C) \to (E', C')$ is a partial function on events $\sigma : E \rightharpoonup E'$ that is local injective:

$$\textit{For all } x \in C, e, e' \in x, \sigma(e) = \sigma(e') \implies e = e'$$

and that extends to a (total) function on configurations:

$$\sigma(x) = x' \textit{ iff } \sigma(|x|) = |x'| \textit{ and } \forall e, e' \in \mathrm{dom}(\sigma), \sigma(e) \leq_{x'} \sigma(e') \iff e \leq_x e'$$

We write $\mathcal{F}_1 \cong \mathcal{F}_2$ whenever there is an isomorphism between $\mathcal{F}_1$ and $\mathcal{F}_2$ and we use $\mathbf{0}$ to denote the rigid family with an empty set of events.

Proposition 1. *Rigid families and their morphisms form a category.*

Importantly, the morphisms we employ here differ from the ones introduced by Hayman and Winskel that are defined on configurations and are not required to preserve the order[3].

If precedence is a partial order that is local to a configuration, one may also define a global (partial) order as follows.

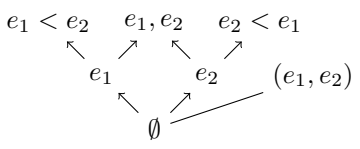

Fig. 1. Example of product

Definition 3 (Causality). *Let $e, e' \in E$ for (E, C) a rigid family. Define $e' < e$ to mean: $\forall x \in C$, if $e, e' \in x$ then $e' <_x e$.*

Rigid families offer a natural notion of *disjoint* causality: i.e an event e_1 is caused by either e_2 or e_3. This type of dependency is a generalisation of Definition 3:

Definition 4 (Disjoint causality). *Let (E, C) a rigid family and $e \in E$, $X \subset E$ such that $e \notin X$. Then X is a disjoint causal set for e, denoted $X < e$ iff the following hold:*

1. **disjointness** $\forall e' \in X$, $\exists x \in C$ *such that* $e' <_x e$ *and* $\forall e'' \in X \setminus e'$, $e'' \not<_x e$.
2. **completeness** $\forall x \in C$, $e \in x \implies \exists e' \in X$ *such that* $e' <_x e$;

In particular $e' < e$ whenever $\{e'\} < e$.

Definition 5 (Concurrency). *Let (E, C) a rigid family and $e, e' \in E$. Define $e \diamond e' \iff \exists x \in C$, $e, e' \in x$ such that $e' \not\leq_x e$ and $e \not\leq_x e'$.*

Note that concurrency has an existential quantifier: two events are concurrent if there exists a configuration in which they are not comparable. On the other hand, causality is universal: it has to hold for all configurations.

[3] We let the reader refer to appendix for details. We also show in appendix how one can compile an event structure from rigid families and *vice versa*. Importantly, the category of Definition 2 and the category of event structures are not equivalent.

Definition 6 (Operations on rigid families). *Let* $E^\star = E \cup \{\star\}$.

1. **Product** *Let* $\star$ *denote* undefined *for a partial function. Define* $(E, C) = (E_1, C_1) \times (E_2, C_2)$ *where* $E = E_1 \times_\star E_2$ *is the product in the category of sets and partial functions with the projections* $\sigma_1 : E \to E_1^\star$, $\sigma_2 : E \to E_2^\star$. *Define the projections* $\pi_1 : (E, C) \to (E_1, C_1)$, $\pi_2 : (E, C) \to (E_2, C_2)$ *and* $x \in C$ *such that the following hold:*
   - x *is a partial order with* $|x| \in \mathcal{P}(E)$;
   - $\pi_1(e) = \sigma_1(e)$ *and* $\pi_2(e) = \sigma_2(e)$;
   - $\pi_1(x) \in C_1$ *and* $\pi_2(x) \in C_2$;
   - $\forall e, e' \in x$, *if* $\pi_1(e) = \pi_1(e') \neq \star$ *and* $\pi_2(e) = \pi_2(e') \neq \star$ *then* $e = e'$.
   - $\forall e, e' \in x$ *such that* $e, e' \in \mathrm{dom}(x)$, $e <_x e' \iff \pi_1(e) <_{\pi_1(x)} \pi_1(e')$ *and* $\pi_2(e) <_{\pi_2(x)} \pi_2(e')$.
   - $\forall y \subseteq x$ *we have that* $\pi_1(y) \in C_1$ *and* $\pi_2(y) \in C_2$.
2. **Restriction** *Define the restriction of an upward closed set of configurations* $X \subseteq C$ *as* $(E, C) \restriction X = (\cup C', C')$ *with* $C' = C \setminus X$. *We equip the operation with a projection* $\pi : (E, C) \restriction X \to (E, C)$ *such that* π *is the identity on events.*
3. **Prefix** *Define* $e.(E, C) = (e \cup E, C' \cup \emptyset)$, *for* $e \notin E$ *where*

$$x' \in C' \iff x' = \big(\{e <_{x'} e' \mid \forall e' \in x\} \cup x\big) \text{ for some } x \in C.$$

   Let $\pi : e.(E, C) \to (E, C)$ *the projection such that* $\pi(e)$ *is undefined and* π *is the identity on the rest of the events.*

Example 1. We obtain the rigid family in Figure 1 for the product of $(\emptyset \prec \{e_1\})$ and $(\emptyset \prec \{e_2\})$.

Proposition 2. *The following properties hold:*

1. $\mathcal{F}_1 \times \mathcal{F}_2$ *is the cartesian product in the category of rigid families.*
2. $\mathcal{F} \restriction X$ *is a rigid family with the projection* $\pi : \mathcal{F} \restriction X \to \mathcal{F}$ *a morphism.*
3. $e.\mathcal{F}$ *is a rigid family with the projection* $\pi : e.\mathcal{F} \to \mathcal{F}$ *a morphism.*

The following proposition shows that the prefix operation adds event e *before* any other event in the family.

Proposition 3. *Let* $e.(E, C) = (E', C')$. $\forall e' \in E'$, $e \leq e'$.

3 Rigid families for CCS

Criterion 1 (Compositional interpretation) *The interpretation of a process should be given by composing the interpretations of its subprocesses.*

We conceived our category of rigid family as model for a large class of concurrent languages, including the π-calculus. In order to illustrate how this should work, we begin by tuning our formalism to finite CCS [17]. We proceed in a similar, yet more technical, manner in section 4 to model the π-calculus.

As it is standard in causal models for concurrency [21], product of rigid families (Definition 6, Equation 1) essentially creates all possible pairs of events that respect the rigidity constraint imposed by the morphisms of our category. One then needs to prune out the pairs of events that do not correspond to legitimate synchronisations. We prove in subsection 3.2 the correspondence with CCS. For simplicity we do not deal with recursion in this paper and let the reader refer to the appendix for a full treatment of non finitary CCS.

3.1 Definitions

Let N be a set of *names* $N = \{a, b, c, \dots\}$, $\overline{N}$ a set of *co-names* $\overline{N} = \{\overline{a}, \overline{b}, \overline{c}, \dots\}$. The function $\overline{[\cdot]} : N \to \overline{N}$ is a bijection, whose inverse is also denoted by $\overline{[\cdot]}$ so that $\overline{\overline{a}} = a$. Let L be the set of event labels defined by the following grammar:

$$\alpha, \beta ::= a \mid \overline{a} \mid (\alpha, \beta)$$

The pairs of labels are globally denoted by τ. We say that an event is *partial* if it is labelled by a name or a co-name. It represents a possible interaction with the context.

Definition 7 (Labelled rigid families).
A labelled rigid family $\mathcal{F} = (E, C, \ell, \mathsf{P})$ is a rigid family equipped with a distinguished set of names P (the private names of $\mathcal{F}$) and a labelling function $\ell : E \to L$.

We use labels to determine which events cannot occur in a computation:

Definition 8 (Disallowed events). *Let $\mathcal{F} = (E, C, \ell, \mathsf{P})$ and $e \in E$. We say that $\ell(e)$ is* disallowed *if one of the following properties holds:*

1. **[type mismatch]** *$\ell(e) = (\alpha, \beta)$ with $\alpha \notin N \cup \overline{N}$ or $\overline{\alpha} \neq \beta$;*
2. **[private name]** *($\ell(e) = a \in N$ or $\ell(e) = \overline{a} \in \overline{N}$) and $a \in \mathsf{P}$;*

A synchronisation event may only occur between complementary partial events (Equation 1) and partial events may not use a private name (Equation 2).

Definition 9 (Dynamic label). *Define the* dynamic label *of an event as $\hat{\ell}(e) = \ell(e)$ if $\ell(e)$ is allowed and $\perp$ otherwise.*

We extend now the operations of Definition 6 in order to take labels into account.

Definition 10 (Operations on labelled rigid families).

1. **Restriction of a name** *Let $a \notin \mathsf{P}$. Then $(E, C, \ell, \mathsf{P}) \restriction a = (E, C, \ell, \mathsf{P} \cup \{a\}) \restriction X$, where $x \in X$ iff $\exists e \in x$ such that $\hat{\ell}(e) = \perp$.*
2. **Prefix** *Define $\alpha.(E, C, \ell, \mathsf{P}) = (E', C', \ell', \mathsf{P})$ where, for some $e \notin E$, $e.(E, C) = (E', C')$ and $\ell'(e) = \alpha$ and $\ell'(e') = \ell(e')$ for $e' \neq e$.*

3. **Product** *Let* $(E, C) = (E_1, C_1) \times (E_2, C_2)$ *and* π_1, π_2 *the projections* $\pi_i : (E, C) \to (E_i, C_i)$. *Then*

$$(E_1, C_1, \ell_1, \mathsf{P}_1) \times (E_2, C_2, \ell_2, \mathsf{P}_2) = (E, C, \ell, \mathsf{P}_1 \cup \mathsf{P}_2)$$

where $\ell(e) = \begin{cases} \ell_i(\pi_i(e)) & \text{if } \pi_{3-i}(e) = \star \\ \big(\ell_1(\pi_1(e)), \ell_2(\pi_2(e_2))\big) & \text{otherwise} \end{cases}$

4. **Parallel composition**

$$(E_1, C_1, \ell_1, \mathsf{P}_1) \mid (E_2, C_2, \ell_2, \mathsf{P}_2) = (E_1, C_1, \ell_1, \mathsf{P}_1) \times (E_2, C_2, \ell_2, \mathsf{P}_2) \upharpoonright X$$

where $x \in X$ *iff* $\exists e \in x$ *such that* $\hat{\ell}(e) = \bot$.

Definition 11 (Sound rigid family). $\mathcal{F}$ *is* sound *iff* $\forall x \in \mathcal{F}$, $\forall e \in x$, $\hat{\ell}(e) \neq \bot$.

Proposition 4. *Let* $\mathcal{F}_1 = (E_1, C_1, \ell_1, \mathsf{P}_1)$, $\mathcal{F}_2 = (E_2, C_2, \ell_2, \mathsf{P}_2)$ *sound rigid families such that* $\mathsf{P}_1 \cap \mathsf{P}_2 = \emptyset$ *and let* α *a name such that* $\alpha \notin \mathsf{P}_1$. *Then* $\alpha.\mathcal{F}_1$, $\mathcal{F}_1 \upharpoonright a$ *and* $\mathcal{F}_1 \mid \mathcal{F}_2$ *are sound rigid families.*

3.2 Operational correspondence with CCS

Criterion 2 (Sound interpretation) *The interpretation of a process can be equipped with an operational semantics that corresponds to the natural reduction semantics of the process.*

To show the correspondence with the operational semantics of CCS, we need to define a notion of *transition* on rigid families. Intuitively, a computation step consists in triggering a single-event computation $\{e\}$ that belongs to the set of configurations. Once e is consumed, the events in conflict with e are eliminated. The remaining configurations are those that are "above" the configuration $\{e\}$.

Definition 12 (Transitions on rigid families).
Let $(E, C, \ell, \mathsf{P})/e = (E', C', \ell', \mathsf{P})$, *for* $\{e\} \in C$, *be the rigid family obtained after the occurence of event* e *and defined as follows:*

- $x' \in C' \iff x' = x \setminus \{e\}$, *for* $\{e\} \preceq x \in C$;
- $e' \in E' \iff \exists x \in C, \{e\} \preceq x$ *and* $e' \in x$;

For all rigid family $\mathcal{F} = (E, C, \ell, \mathsf{P})$ with $\{e\} \in C$, note that $\mathcal{F}/e$ is also a labelled rigid family. Now consider (finite) CCS terms defined by the grammar below:

$$P, Q ::= (P|Q) \mid a.P \mid \overline{a}.P \mid P\backslash a \mid 0$$

As usual, occurrences of a and $\overline{a}$ in $P\backslash a$ are bound. For simplicity, we assume that all bound occurrences of names are kept distinct from each other and from free occurrences.

The interpretation of a CCS process as a rigid family is defined by induction on the structure of a term:

$$[\![\alpha.P]\!] = \alpha.[\![P]\!] \quad [\![P|Q]\!] = [\![P]\!]|[\![Q]\!] \quad [\![P\backslash a]\!] = [\![P]\!] \upharpoonright a \quad [\![0]\!] = \mathbf{0}$$

Lemma 1. *Let P a process and $[\![P]\!] = (E, C, \ell, \mathsf{P})$ its interpretation.*

1. *$\forall \alpha$, P' such that $P \xrightarrow{\alpha} P'$, $\exists e \in E$ such that $\ell(e) = \alpha$ and $[\![P]\!]/e \cong [\![P']\!]$;*
2. *$\forall e \in E$, $\{e\} \in C$, $\exists P'$ such that $P \xrightarrow{\ell(e)} P'$ and $[\![P]\!]/e \cong [\![P']\!]$.*

A direct corollary of Lemma 1 is that a process and its encoding can simulate each others reductions.

Theorem 1 (Operational correspondence with CCS). *Let P a process and $[\![P]\!] = (E, C, \ell, \mathsf{P})$ its encoding.*

1. *$\forall P'$ such that $P \xrightarrow{\tau} P'$, $\exists \{e\} \in C$ closed such that $[\![P]\!]/e \cong [\![P']\!]$;*
2. *$\forall e \in E$, $\{e\} \in C$ closed, $\exists P'$ such that $P \xrightarrow{\tau} P'$ and $[\![P]\!]/e \cong [\![P']\!]$.*

3.3 Causality and concurrency in CCS

A term in CCS can compose with a *context* and then exhibit more behaviours. We will show below a property that says that precedence in the semantics of a term can be seen as an abstraction of the causality that appears when the term is put into a context.

In the interpretation of a CCS term, if we have a configuration x and two concurrent events in x, we also have the configuration y with the same events and the same order as x except for the concurrent events that become ordered in y. This is stated formally in the following proposition.

Proposition 5. *Let $[\![P]\!] = (E, C, \ell, \mathsf{P})$. $\forall x \in C$ and $\forall e_1, e_2 \in x$ such that $e_1 \diamond_x e_2$, $\exists y \in C$ such that $|x| = |y|$ and*

1. *y preserves the order in x: $\forall e, e' \in x$, $e \leq_x e' \implies e \leq_y e'$*
2. *x reflects the order in y except for $e_1 <_y e_2$: $\forall e, e' \in y$ such that $\neg(e = e_1 \wedge e' = e_2)$, $e \leq_y e' \implies e \leq_x e'$.*

In CCS we cannot guarantee simultaneity. Two concurrent events (Definition 5) can be observed simultaneously but also in any order. For instance the order $a < b$ induced by the CCS term $a \mid b$ is materialized when the term composes with the context $(\overline{a}.\overline{b} \mid [\cdot])\backslash ab$. Note that this is a property specific to CCS (and the π-calculus in subsection 4.4), but one can encode in rigid families calculi where such a property does not hold.

The causality of Definition 3 is called *structural* in previous models for CCS [3,22]. But we can also express the disjunctive causality of Definition 4 in CCS. Consider the process $a.b \mid \overline{a}$ interpreted in rigid families in Figure 2, where events are replaced by their corresponding labels. The disjunctive causal set for the event labelled b consists of the events labelled a and τ: $X = \{a, \tau\} < b$. Disjunctive causal sets in CCS always consist of conflicting events. However, it is not the case for the disjunctive causal sets of the π-calculus.

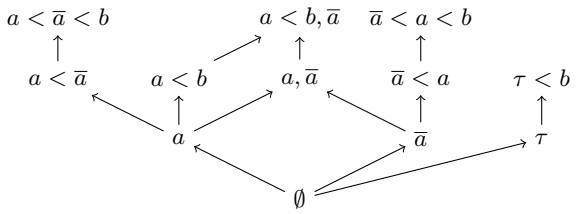

Fig. 2. $a.b \mid \overline{a}$ in rigid families

4 Rigid families for the π-calculus

We show how definitions in section 3 can be adapted to the π-calculus [18]. The treatment of synchronisation labels is more complicated as noted in Ref. [16,9] since names can be substituted during computations. Also, restriction does not necessarily delete events labelled with the private name, due to the phenomenon of *scope extrusion*.

However, in our novel approach, the main difficulty of the encoding resides in the definition of disallowed labels. Given the correct definition, all operations on rigid families for the π-calculus are straightforward extensions from CCS, including the restriction. As in CCS, for simplicity, we do not treat recursion or nondeterministic sum.

4.1 Labels for the π-calculus

We redefine the set L of events labels (see below), in order to handle the labels of the π-calculus. We use α, β to range over L, on which we define the functions subj and obj in the obvious manner:

$$\alpha ::= \overline{b}\langle a\rangle \mid d(c) \mid (\alpha, \beta)$$
$$\mathsf{subj}(\overline{b}\langle a\rangle) = \{b\}\ \ \mathsf{subj}(d(c)) = \{d\}\ \ \mathsf{subj}(\alpha, \beta) = \mathsf{subj}(\alpha) \cup \mathsf{subj}(\beta)$$
$$\mathsf{obj}(\overline{b}\langle a\rangle) = \{a\}\ \ \mathsf{obj}(d(c)) = \{c\}\quad \mathsf{obj}(\alpha, \beta) = \mathsf{obj}(\alpha) \cup \mathsf{obj}(\beta)$$

A labelled rigid family for the π-calculus is defined as in Definition 7, except that the labelling function $\ell : E \to L$ has as codomain the new set L. For a label $\alpha = b(a)$ or $\alpha = \overline{b}\langle a\rangle$, we use the notation $\alpha \in \ell(e)$ if $\ell(e) = (\alpha, \alpha')$, $\ell(e) = (\alpha', \alpha)$ or $\ell(e) = \alpha$. The name b is *binding* in a label α if $\alpha = a(b)$ for some name a. For simplicity we use the *Barendregt convention*: a name b has at most one binding occurrence in all event labels and this occurrence binds all bs in the other event labels.

We call an event e that binds a name b in a configuration an *instantiator* for b. An event e with label $\overline{c}\langle d\rangle$ can thus have two instantiators, one for the subject c and one for the object d. If the event is labelled by a synchronisation $(\ell(e) = (\overline{b}\langle a\rangle, d(c)))$ we can have up to three instantiators (for the names b, a and d). For all e occurring in a configuration x, we write $e' \in \mathsf{inst}^{\mathrm{s}}_x(e)$ and

$e'' = \mathsf{inst}^o_x(e)$ for, respectively, the subject and object instantiator of e. Note that in the interpretation of a π process (respecting the Barendregt convention) as a rigid family in subsection 4.3, it can be proved that $e' <_x e$ and $e'' <_x e$.

4.2 Synchronizations

Let Σ be the set of all name substitutions. The function $\sigma_x : x \to \Sigma$ returns a set with all substitutions generated by synchronisation events in x.

Definition 13 (Substitution). *We define σ_x by induction on x:*

$$\begin{aligned}
\sigma_\emptyset =& \emptyset \\
\sigma_x =& \sigma_{x\setminus e} \text{ if } \ell(e') \neq (d(a), \overline{b}\langle a'\rangle) \\
& \sigma_{x\setminus e} \cup \{a'/a\} \text{ if } \ell(e') = (d(a), \overline{b}\langle a'\rangle) \text{ and } \{a''/a'\} \notin \sigma_{x\setminus e} \\
& \sigma_{x\setminus e} \cup \{a''/a\} \text{ if } \ell(e') = (d(a), \overline{b}\langle a'\rangle) \text{ and } \{a''/a'\} \in \sigma_{x\setminus e}
\end{aligned}$$

Define $\ell_x(e) = \ell(e)\sigma_x$ which applies the substitutions to the label of e.[4]

We can prove that for any configuration x in the interpretation of a π process, σ_x is well defined.

The synchronizations of a configuration x are events $e \in x$ such that $\ell(e) = (b(a), \overline{c}\langle d\rangle)$, for some names a, c, d. We use the configuration-indexed predicate $\tilde{\tau}_x : x \to 2$ to denote events of that sort. The *materialized* synchronisations are synchronization events with the $\ell_x(e) = (a(c), \overline{a}\langle d\rangle)$. The predicate $\tau_x : x \to 2$ is the smallest predicate that holds for such events.

Importantly, one cannot simply screen out all $\tilde{\tau}_x$-events in a rigid family that are not also τ_x-events. They might still become fully fledged synchronisations after composing the family with more context. Yet some pairs of events will never satisfy the τ_x predicate, no matter what operations are applied to the rigid family they belong to. Such events can be identified thanks to their *disallowed* label.

The definitions of events that have disallowed labels are quite cumbersome but essentially consist in an exhaustive characterization of events the label of which proves they can no longer appear in a π-calculus computation (Definition 15 and Definition 16). Such events can only appear after applying the product or the restriction, operations needed to represent the parallel composition and name restriction of the π-calculus. They are therefore detected and removed on the fly (see Definition 10). The reader, uninterested in technical details, may now safely skip to subsection 4.3, having in mind this informal notion of disallowed events.

A $\tilde{\tau}_x$-event can become a materialized τ_x-event when the names used as subject can be *matched*. This is always a possibility if the names are not private, because input prefix operations can instantiate them to a common value. However, when only one of the names is private then a distinct event, occurring beforehand, has to be in charge of leaking the name to the context. We call such event an *extruder*:

[4] We give the formal definition in Appendix subsection A.5.

Definition 14 (Extruder). *An event $e \in x$ is an* extruder *of $e' \in x$ if $e <_x e'$ and $\ell_x(e) = \overline{b}\langle a \rangle$ for some b where a is private and $a \in \mathsf{subj}(\ell_x(e'))$.*

Consider a $\tilde{\tau}_x$-event e occurring in a configuration x. If $a, b \in \mathsf{subj}(\ell_x(e))$ we say that a can *eventually match* b iff $a \notin \mathsf{P}$ and either $b \notin \mathsf{P}$ or if $b \in \mathsf{P}$ then there exists $e', e'' \in x$, $e' = \mathsf{inst}^s_x(e)$ and e'' extruder of e such that $e'' <_x e'$. We write $\mathsf{U}(a, b)$ iff $a = b$ or if a can eventually match b or b can eventually match a.

Definition 15 (Disallowed $\tilde{\tau}_x$-events). *Let $\mathcal{F} = (E, C, \ell, \mathsf{P})$ and $x \in C$, $e \in x$ with $\tilde{\tau}_x(e)$. The label $\lambda = \ell_x(e)$ is* disallowed *if one of the following properties holds:*

1. **[type mismatch]** *$\lambda = (\alpha, \beta)$ and it is not the case that α is input and β output or viceversa;*
2. **[non unifiable labels]** *let $a, b \in \mathsf{subj}(\lambda)$ and either:*
   - *$\neg\, \mathsf{U}(a, b)$*
   - *$\exists e' \in x$ such that $a, b' \in \mathsf{subj}(\ell_x(e'))$ for some b' and $\neg\, \mathsf{U}(b, b')$.*

Condition 1 is straightforward: an output can only synchronize with an input. Condition 2 says that a $\tilde{\tau}_x$-event cannot materialize if the names used in subject position cannot eventually match.

Private names cannot be used to interact with the context. Therefore we disallows partial events that use a private name as a communication channel (i.e. in the subject of the label). However, if the private name a is sent to the context (by an extruder) then a synchronisation on a becomes possible. This is formally stated by the definition below.

Definition 16 (Disallowed partial event). *Let $\mathcal{F} = (E, C, \ell, \mathsf{P})$ and $x \in C$, $e \in x$ with $\neg\tilde{\tau}_x(e)$. The label $\lambda = \ell_x(e)$ is* disallowed *if $a \in \mathsf{subj}(\lambda)$, $a \in \mathsf{P}$ and $\nexists e' \in x$ extruder of e.*

Definition 17 (Dynamic label). *Define the dynamic label of an event as $\hat{\ell}_x(e) = \ell_x(e)$ if $\ell_x(e)$ is allowed and $\perp$ otherwise.*

We have the same operations as in Definition 10 but applied to the set of labels of the π-calculus and using the dynamic labels of Definition 17.

4.3 Operational correspondence with the π calculus

In this section we show the operational correspondence between processes in π-calculus and their encoding in rigid families. We assume the reader is familiar with the π-calculus [18,20]. For the sake of simplicity we use a restricted version of π-calculus defined by the grammar below:

$$P ::= (P|P) \mid \overline{b}\langle a \rangle.P \mid d(c).P \mid P\backslash a \mid 0$$

The restriction of the private name a is denoted as in CCS, to highlight that in their interpretation in rigid families is similar. We use a late LTS for the π-calculus recalled in the appendix. Similarly to subsection 3.2 we encode a process into a rigid family as follows:

$$[\![\alpha.P]\!] = \alpha.[\![P]\!] \quad [\![P|Q]\!] = [\![P]\!]|[\![Q]\!] \quad [\![P\backslash a]\!] = [\![P]\!] \upharpoonright a \quad [\![0]\!] = \mathbf{0}$$

We now revisit definition 12 since transitions on rigid families with dynamic labels need to apply substitutions on the fly.

Definition 18 (Transitions on rigid families).
Let $(E, C, \ell, \mathsf{P})/e = (E', C', \ell', \mathsf{P})$, for $\{e\} \in C$, be the rigid family obtained after the occurence of event e and defined as follows:

- $x' \in C' \iff x' = x \setminus \{e\}$, *for* $\{e\} \preceq x \in C$;
- $E' = \cup|x'|$, *for all* $x' \in C'$;
- *if* $\ell(e) = (\bar{b}\langle a\rangle, c(d))$ *then* $\ell'(e) = \ell(e)\{a/d\}$; *otherwise* $\ell' = \ell$.

In a rigid family we have events that do not have an operational correspondence, but are necessary for compositionality. These are the events for which the predicate $\tilde{\tau}_x(e)$ holds but $\tau_x(e)$ does not. We ignore them when showing the operational correspondence.

Definition 19 (Complete and closed configurations). *A configuration x in a rigid family (E, C, ℓ, P) is* complete *if $\forall e \in x$, $\tilde{\tau}_x(e) \implies \tau_x(e)$. We say that x is* closed *if $\forall e \in x$, $\tau_x(e)$ holds.*

Remark that for minimal events (i.e. e such that $\{e\}$ is a configuration) $\ell_{\{e\}}(e) = \ell(e)$.

Lemma 2. *Let P a process and $[\![P]\!] = (E, C, \ell, \mathsf{P})$ its encoding.*

1. *$\forall \alpha$, P' such that $P \xrightarrow{\alpha} P'$, $\exists e \in E$ such that $\ell(e) = \alpha$ and $[\![P]\!]/e \cong [\![P']\!]$;*
2. *$\forall e \in E$, $\{e\} \in C$ complete, $\exists P'$ such that $P \xrightarrow{\ell(e)} P'$ and $[\![P]\!]/e \cong [\![P']\!]$.*

Proof (sketch).

1. We proceed by induction on the derivation of the rule $P \xrightarrow{\alpha} P'$. We consider the following cases:

$$\textbf{In/Out}\frac{}{\alpha.P \xrightarrow{\alpha} P} \quad \textbf{Com}\frac{P \xrightarrow{\bar{b}\langle a\rangle} P' \quad Q \xrightarrow{b(c)} Q'}{P|Q \xrightarrow{\tau} P'|Q'\{a/c\}} \quad \textbf{Restr}\frac{P \xrightarrow{\alpha} P'}{P\backslash a \xrightarrow{\alpha} P'\backslash a}\, a \notin \alpha$$

   - Rule In/Out: let $[\![\alpha.P]\!] = \alpha.[\![P]\!]$ hence we have to show that

$$[\![P]\!] \cong (\alpha.[\![P]\!])/e, \text{ where } \ell(e) = \alpha. \tag{1}$$

   - Rule Com: we have $[\![P|Q]\!] = ([\![P]\!] \times [\![Q]\!]) \upharpoonright X$. By induction $[\![P']\!] \cong [\![P]\!]/e_1$ where $\ell(e_1) = \bar{b}\langle a\rangle$ and $[\![Q']\!] \cong [\![Q]\!]/e_2$ where $\ell(e_2) = b(c)$. Then $\{e_1\} \in [\![P]\!]$ and $\{e_2\} \in [\![Q]\!]$ it implies that there exists $\{e\} \in [\![P]\!] \times [\![Q]\!]$ such that $\pi_1(e) = e_1$ and $\pi_2(e) = e_2$. Hence we have to show that

$$\Big(([\![P]\!]/e_1 \times [\![Q]\!]/e_2) \upharpoonright X'\Big)\{a/d\} \cong \big(([\![P]\!] \times [\![Q]\!]) \upharpoonright X\big)/e$$

     where $\mathcal{F}\{a/d\}$ replaces d with a in all labels.

– Rule Restr: by induction we have that $[\![P]\!]/e \cong [\![P']\!]$, where $\ell(e) = \alpha$. We have then to show that $\{e\} \in [\![P\backslash a]\!]$ and

$$([\![P]\!]/e) \upharpoonright a \cong ([\![P]\!] \upharpoonright a)/e \tag{2}$$

2. We proceed by structural induction on P. Consider the following cases as example:
   – $P = \alpha.P'$ then $[\![P]\!] = [\![\alpha.P']\!] = \alpha.[\![P']\!]$. There exists only one singleton configuration $\{e\} \in \alpha.[\![P']\!]$, where $\ell(e) = \alpha$. We have that $\alpha.P' \xrightarrow{\alpha} P'$ hence $[\![P']\!] \cong (\alpha.[\![P']\!])/e$, which follows from Equation 1.
   – $P = P'\backslash a$ then $[\![P]\!] = [\![P']\!] \upharpoonright a$. We have that $\forall \{e\} \in [\![P]\!]$, $a \notin \mathsf{subj}(\ell(e))$ and $\{e\} \in [\![P]\!]$.
   Consider the subcases where either $a \notin \ell(e)$ or $\tilde{\tau}_x(e)$ holds.
   Then $P'\backslash a \xrightarrow{\ell(e)} P''\backslash a$ and $P' \xrightarrow{\ell(e)} P''$. By induction $[\![P'']\!] \cong [\![P']\!]/e$. Hence we have to show that $[\![P'']\!] \upharpoonright a \cong ([\![P']\!] \upharpoonright X_a)/e$ that is $([\![P']\!]/e) \upharpoonright X_a \cong ([\![P']\!] \upharpoonright X_a)/e$ which follows from Equation 2.

A direct consequence of Lemma 2 is that, as in CCS, a process and its encoding can simulate each others reductions.

Theorem 2 (Operational correspondance with the π calculus). *Let P a process and $[\![P]\!] = (E, C, \ell, \mathsf{P})$ its encoding.*

1. *$\forall P'$ such that $P \xrightarrow{\tau} P'$, $\exists \{e\} \in C$ closed such that $[\![P]\!]/e \cong [\![P']\!]$;*
2. *$\forall e \in E$, $\{e\} \in C$ closed, $\exists P'$ such that $P \xrightarrow{\tau} P'$ and $[\![P]\!]/e \cong [\![P']\!]$.*

4.4 Causality and concurrency in the π-calculus

Proposition 5 extends to the π-calculus.

The disjoint causal sets in the π-calculus, capture the causality induced by the prefix operator (as in CCS), but also the causality induced by the restriction of private names. Consider as an example the process $(\overline{b}\langle a\rangle \mid \overline{c}\langle a\rangle \mid a)\backslash a$ with its encoding in rigid families in Figure 3, where events are replaced by their labels. The disjoint causal set for the event labelled a consists of the events labelled $\overline{b}\langle a\rangle$ and $\overline{c}\langle a\rangle$, that is $\{\overline{b}\langle a\rangle, \overline{c}\langle a\rangle\} < a$. Indeed the event labelled a cannot appear by itself: either $\overline{b}\langle a\rangle$ or $\overline{c}\langle a\rangle$ precedes it. However in rigid families disjunctive causality [9] is not ambiguous: in every configuration the order on the events is fixed.

5 Advanced criteria

In the previous section we have presented an interpretation that is compositional and sound both for CCS and the π-calculus. We propose in this section a few additional notions of correctness that our semantics enjoys. These criteria are of particular interest in the π-calculus, as previous causal models do not satisfy them. Hence we only formally prove them for the π-calculus, but they can be shown to hold for CCS as well.

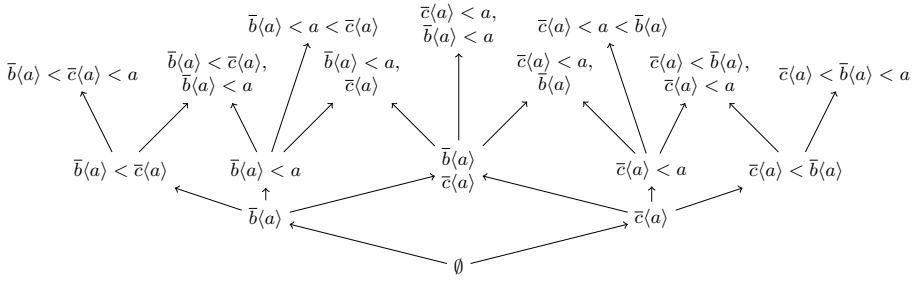

Fig. 3. $(\overline{b}\langle a\rangle \mid \overline{c}\langle a\rangle \mid a)\backslash a$ in rigid families

5.1 Realisability

Criterion 3 (Realisable interpretation) *Every labelled run should represent a possible execution.*

A labelled trace of an open system can be closed if the right context composes with the open term. In this case we say that the trace is realisable. This can be seen somehow as the dual to soundness. If soundness means that the model has *enough* labelled events and runs, we also require that it has not *too many* of them. In order to formalise this criterion, we need to use the notion of *context*. A context C in the π-calculus is defined as a process with a hole:

$$C[\cdot] := [\cdot] \parallel \alpha.C[\cdot] \parallel P|C[\cdot] \parallel C[\cdot]\backslash a$$

We do not know how to define a notion of context for rigid families in general, as it is not clear what is a rigid family with a hole. However, if a structure $\mathcal{F}$ has an operational meaning (i.e. $\exists P$ a process such that $\mathcal{F} = [\![P]\!]$) we can use a π-calculus context $C[\cdot]$ to define a new term $[\![C[P]]\!]$.

When analysing the reductions of a process in context, we need to know the contribution the process and the context have in the reduction. To this aim we associate to the context $C[\cdot]$ instantiated by a process P a projection morphism $\pi_{C,P} : [\![C[P]]\!] \to [\![P]\!]$ that can retrieve the parts of a configuration in $[\![C[P]]\!]$ that belong to $[\![P]\!]$.

Definition 20. *Let $C[\cdot]$ a π-calculus context, and P a process. The* projection $\pi_{C,P} : [\![C[P]]\!] \to [\![P]\!]$ *is inductively defined on the structure of C as follows:*

- $\pi_{C,P} : [\![\alpha.C'[P]]\!] \to [\![P]\!]$ *is defined as* $\pi_{C,P}(e) = \pi_{C',P}(e)$*;*
- $\pi_{C,P} : [\![C'[P]|P']\!] \to [\![P]\!]$ *is defined as* $\pi_{C,P}(e) = \pi_{C',P}(\pi_1(e))$*, where* $\pi_1 : [\![C'[P]|P']\!] \to [\![C'[P]]\!]$ *is the projection morphism defined by the product;*
- $\pi_{C,P} : [\![C'[P]\backslash a]\!] \to [\![P]\!]$ *defined as* $\pi_{C,P}(e) = \pi_{C',P}(e)$.

One can easily verify, by case analysis, that the projection $\pi_{C,P} : [\![C[P]]\!] \to [\![P]\!]$ is a morphism. We naturally extend $\pi_{C,P}$ to configurations.

We can now prove our first criterion: every partial labelled configuration can be "closed".

Theorem 3. *$\forall x \in [\![P]\!]$, there exists C a context for $[\![P]\!]$ and $z \in [\![C[P]]\!]$ closed such that $\pi_{C,P}(z) = x$.*

Proof (sketch). We proceed in several steps:

1. We show that for $x \in [\![P]\!]$ there exists a context $C_1 = \alpha_1 \cdots \alpha_n.[\cdot]$ and $x_1 \in [\![C_1[P]]\!]$ such that
   - $\pi_{C_1,P}(x_1) = x$ and
   - $\forall e \in x_1$, if $\tilde{\tau}_{x_1}(e)$ then $\forall b \in \mathsf{subj}(\ell_{x_1}(e))$ we have that $b \notin \mathsf{P} \implies \exists e_1 \in x_1, e_1 \in \mathsf{inst}^s_{x_1}(e)$ such that $\ell(e_1) = d'(b)$.
2. Define a *precontext* as a multiset of labels such that $b(a) \in \chi \implies \nexists c(a) \in \chi$. For $x_1 \in [\![P_1]\!]$ we define a precontext and a function $f : \chi \to x_1$ associating a label to any partial event. Intuitively for every open or partial event $e \in x_1$ we associate a label $\alpha \in \chi$ that "closes" the event. Let ς be a set of substitutions generated by x_1 *and* χ.
   We ask that given x_1, the precontext χ and the total and injective function $f : \chi \to x_1$ satisfy the following:
   - $\forall e \in x_1$ with $\ell_{x_1}(e) = (\overline{b}\langle a\rangle, c(d))$ then $\ell_\chi(e) = (\overline{b}\langle a\rangle, b(d))$;
   - $\forall e \in x_1$ with $\ell_{x_1}(e) = \overline{b}\langle a\rangle$ or $\ell_{x_1}(e) = b(d)$ there exists $\alpha \in \chi$, $\alpha = b'(a')$ or $\alpha = \overline{b'}\langle a'\rangle$ respectively, such that $f(\alpha) = e$ and $b'\varsigma = b$;
   - $\forall \alpha_1, \alpha_2 \in \chi$, such that $\alpha_1 = b(a)$, $a = \mathsf{subj}(\alpha_2) \iff f(\alpha_1) <_{x_1} f(\alpha_2)$ and there is $a \in \mathsf{obj}(\ell_{x_1}(f(\alpha_1)))$ private.

   In the above we have that $\ell_\chi(e) = \ell(e)\varsigma(e)$.
3. Let $x_1 \in [\![P_1]\!]$ with χ a precontext and $f : \chi \to x_1$ a total function as defined above. We construct a process P_2 from the precontext χ such that
   - there are no private names in P_2: $(\backslash a) \notin P_2$, for any name a;
   - $\alpha \in P_2 \iff \alpha \in \chi$;
   - $\alpha_1 \cdots \alpha_n.0 \in P_2 \iff \exists a \in \mathsf{obj}(\ell_{x_1}(e_1)), a \in \mathsf{P}$ and $e_n <_{x_1} e_i, a \in \mathsf{subj}(\ell_{x_1}(e_i))$ for $f(\alpha_1) = e_1$ and $f(\alpha_i) = e_i$, with $i \in \{2, n\}$.

   The context required by the theorem is $C = P_2 \mid [\cdot]$. The conditions above guarantee that we construct the context from the precontext and that the sequential operator are only used for an extrusion from P (and the instantiation in P_2) of a private name.
4. We show that $\exists x_2 \in [\![P_2]\!]$ and $g : x_2 \to x_1$ a total, injective function such that

$$\begin{aligned} \alpha \in P_2 &\iff e \in x_2, \ell(e) = \alpha \\ g(e_2) = e_1 &\iff f(\ell(e_2)) = e_1 \\ e_2 <_{x_2} e_2' &\iff g(e_2) <_{x_1} g(e_1) \end{aligned}$$

   Intuitively, x_2 is the configuration that composes with x_1 in order to produce the closed configuration z. We have that x_2 is maximal in $[\![P_2]\!]$, hence it contains all labels in χ. The conditions above ensure that the function g keeps the correspondence between partial events in x_1 and their 'future' synchronisation partners in x_2.

5. Let $x_1 \in [\![P_1]\!]$ and $x_2 \in [\![P_2]\!]$ defined above. We have that $[\![P_1|P_2]\!] = \big([\![P_1]\!] \times [\![P_2]\!]\big) \restriction X$ and π_1, π_2 the projections. Denote $[\![P_1|P_2]\!] = (E, C)$. We show that $\exists z \in [\![P_1|P_2]\!]$ closed with $\pi_1(z) = x_1$ and such that $z \notin X$.

Using Theorem 3 we have the following straightforward corollary, that says that any context has to preserve and reflect the concurrency relation on events, and consequently the precedence between events. It follows from the preservation and reflection of order by the morphisms.

Corollary 1. *$\forall x \in [\![P]\!]$ and $\forall C$ context for $[\![P]\!]$ such that $z \in [\![C[P]]\!]$ and $\pi_{C,P}(z) = x$ we have that $e_1 \diamond_x e_2 \iff \pi^{-1}_{C,P}(e_1) \diamond_z \pi^{-1}_{C,P}(e_2)$.*

5.2 Structural congruence

Criterion 4 (Denotational interpretation) *The interpretation should be invariant for structural congruence.*

Corollary 1 says that the interpretation does not contain too much concurrency. We also would like to prove that the interpretation contains *enough* concurrency, or, dually, that it does not have *too much* causality: specifically we require that the restriction operator does not introduce too much causality. Surprisingly this is obtained by simply requiring that the semantics preserves structural congruence. The interesting case is the scope extension rule:

Theorem 4 (Preservation of structural congruence).

$$[\![(P)\backslash a \mid Q]\!] \cong [\![(P \mid Q)\backslash a]\!] \text{ if } a \notin \mathrm{fn}(Q)$$

Proof (sketch). We show that

$$\mathcal{F}_1 = \big([\![P]\!] \restriction X_1 \times [\![Q]\!]\big) \restriction X_2 \cong \big([\![P]\!] \times [\![Q]\!]\big) \restriction X_3 = \mathcal{F}_2.$$

where $\mathcal{F}_1 = (E_1, C_1, \ell_1, \mathsf{P}_1)$ and $\mathcal{F}_2 = (E_2, C_2, \ell_2, \mathsf{P}_2)$. Let us denote $\pi_{1,P} : \mathcal{F}_1 \to [\![P]\!]$ and $\pi_{1,Q} : \mathcal{F}_1 \to [\![Q]\!]$ and similarly for $\pi_{2,P}$, $\pi_{2,Q}$. Define a bijection on events as follows $\iota(e_1) = e_2 \iff \pi_{1,P}(e_1) = \pi_{2,P}(e_2)$ and $\pi_{1,Q}(e_1) = \pi_{2,Q}(e_2)$. To show it is an isomorphism we show that $x_1 \in \mathcal{F}_1 \iff x_2 \in \mathcal{F}_2$, where $\iota(x_1) = x_2$.

To better understand the importance of this criterion, consider the causal models presented in [2]. In those models, there is a tight correspondence with the standard transition semantics of the π-calculus. In particular, the first output that extrudes a name has a different label than all subsequent outputs of that name, and moreover it precedes them in the causal relation. If we had made a similar choice here, in the process $P = (\overline{b}\langle a\rangle \mid \overline{c}\langle a\rangle)\backslash a$, we would have only configurations where one of the output would precede the other. By the way parallel composition is defined, this would imply that in $P \mid (b(x) \mid c(y))$ the τ transitions are causally related. However this process is structurally congruent to $(\overline{b}\langle a\rangle \mid \overline{c}\langle a\rangle \mid b(x) \mid c(y))\backslash a$, where the τ transitions are concurrent. Thus Theorem 4 would fail. Though the causal model of [2] is defined on transition systems one can represent the causality relation induced in a rigid family as in Figure 4. We can see then that the two processes $(\overline{b}\langle a\rangle \mid \overline{c}\langle a\rangle)\backslash a| \ (b(x) \mid c(y))$ and $(\overline{b}\langle a\rangle \mid \overline{c}\langle a\rangle \mid b(x) \mid c(y))\backslash a$ have different interpretations.

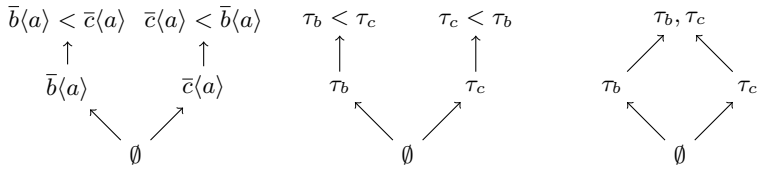

Fig. 4. $(\overline{b}\langle a\rangle \mid \overline{c}\langle a\rangle)\backslash a$, $(\overline{b}\langle a\rangle \mid \overline{c}\langle a\rangle)\backslash a|\ (b(x) \mid c(y))$ and $(\overline{b}\langle a\rangle \mid \overline{c}\langle a\rangle \mid b(x) \mid c(y))\backslash a$ in the model of [2]

5.3 Reversibility and Stability

In the operational setting, reversible variants for CCS [11] and the π-calculus [10] have been studied. We conjecture that rigid families are suitable models for the reversible π- calculus, but leave this as future work. Instead we show, intuitively, why rigid families are a good fit for reversibility.

Reversible calculi allow actions to backtrack in a different order then they appeared in the forward computation. The only constraint is that the causal order between events is respected: we cannot undo the cause before the effect. Configuration structures and rigid families are a suited for reversible calculi as the (rigid) inclusion between sets dictates the allowed forward and backward transitions. Then it is important that in the closed term we can backtrack on any path allowed by the forward computation. We can generalise the realisability criterion (Theorem 3) to the reversible setting: there exists a context that closes a configuration and that preserves all possible paths leading to it.

In previous works [22] this condition is called *stability* and is defined on the domain underlying the rigid families. Intuitively stability means that in any configuration one can deduce a partial order on events. Notably, the semantics proposed in [9] does not satisfy it.

Consider the process $(\overline{b}\langle a\rangle \mid \overline{c}\langle a\rangle \mid a(d))\backslash a$ with the rigid family depicted in Figure 3. Its representation in [9] is represented in Figure 5. In this process a is private and it cannot be used as a communication channel. The context has first to receive the name from one of its extruders: on channel b or on channel c. This type of *disjunctive* causality is what the model of [9] depicts. In the top configuration the extruder of a is either $\overline{b}\langle a\rangle$ or $\overline{c}\langle a\rangle$. However in a closed term we can never express this type of disjunctive causality. We can either close the term with a process of the form $b(a').\overline{a'}\langle\rangle$ or $c(a'').\overline{a''}\langle\rangle$. Hence there is no context which can close such a configuration. In our model, instead, in any configuration that has disjunctive causality the order of occurrence between events is fixed.

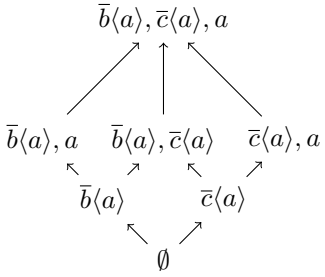

Fig. 5. The configuration structures of $(\overline{b}\langle a\rangle \mid \overline{c}\langle a\rangle \mid a(d))\backslash a$ in [9]

6 Conclusions

We presented a novel causal semantics for concurrent computations. Importantly, we exhibit correction criteria that differentiate our model from others.We have stated that a correct causal model should be:

1. *Compositional:* we have defined a category of rigid families such that the encoding of a term is given by a categorical operation on the encoding of its subterms.
2. *Realisable:* each configuration in the interpretation of a process is realisable and the precedence in the closed system is the same as in the open one.
3. *Sound:* we showed an operational correspondence with the reduction semantics of CCS and the π-calculus.
4. *Denotational:* the rules of structural congruence are preserved by the causal semantics.

The first two correction criteria can be seen as *internal coherence* results: open traces can compose to form correct closed executions. *External coherence* criteria relate rigid families to other models for concurrency. In this paper we used the model to give interpretations to CCS and π-calculus processes that satisfy the third and fourth correction criteria. As future work, we plan to show a correspondence between reversible π-calculus [10] and its encoding in rigid families. The correction criteria have then to hold in a reversible setting as well. As we showed in subsection 5.3 this is particularly interesting for the realisability criterion.

In the π-calculus the input prefix plays the double role of instantiator and of structural predecessor. We can interpret in rigid families a calculus without prefix precedence [24] and we conjecture that the partial orders would then characterise the information flow rather then the temporal precedence.

Equivalence relations defined on previous causal model for CCS have a correspondence in reversible CCS [1]. We plan to show as future work that such equivalence relations can be defined on rigid families and possibly correspond to bisimulations in the reversible π-calculus.

References

1. Clement Aubert and Ioana Cristescu. Reversible barbed congruence on configuration structures. In *8th ICE, Satellite workshop of DisCoTec 2015*, 2015.
2. Michele Boreale and Davide Sangiorgi. A fully abstract semantics for causality in the π-calculus. *Acta Inf.*, 35(5):353–400, 1998.
3. Gérard Boudol and Ilaria Castellani. Flow models of distributed computations: Three equivalent semantics for CCS. *Inf. Comput.*, 114(2):247–314, 1994.
4. Roberto Bruni, Hernán Melgratti, and Ugo Montanari. Event structure semantics for nominal calculi. CONCUR'06, pages 295–309. Springer-Verlag, 2006.
5. Nadia Busi and Roberto Gorrieri. A petri net semantics for π-calculus. CONCUR '95, pages 145–159. Springer-Verlag, 1995.

6. Simon Castellan, Jonathan Hayman, Marc Lasson, and Glynn Winskel. Strategies as concurrent processes. *Electron. Notes Theor. Comput. Sci.*, 308(0):87 – 107, 2014. Proceedings of the 30th Conference on the Mathematical Foundations of Programming Semantics (MFPS XXX).
7. Gian Luca Cattani and Peter Sewell. Models for name-passing processes: Interleaving and causal. *Inf. Comput.*, 190(2):136–178, May 2004.
8. Gian Luca Cattani and Glynn Winskel. Presheaf models for ccs-like languages. *Theor. Comput. Sci.*, 300(1–3):47 – 89, 2003.
9. Silvia Crafa, Daniele Varacca, and Nobuko Yoshida. Event structure semantics of parallel extrusion in the π-calculus. In *Foundations of Software Science and Computational Structures*, volume 7213 of *LNCS*, pages 225–239. 2012.
10. Ioana Cristescu, Jean Krivine, and Daniele Varacca. A compositional semantics for the reversible π-calculus. LICS'13, pages 388–397. IEEE Computer Society.
11. Vincent Danos and Jean Krivine. Reversible communicating systems. In *CONCUR 2004 - Concurrency Theory, 15th International Conference, Proceedings*, volume 3170 of *LNCS*, pages 292–307, 2004.
12. Pierpaolo Degano and Corrado Priami. Non-interleaving semantics for mobile processes. *Theor. Comput. Sci.*, 216(1-2):237–270, March 1999.
13. Clovis Eberhart, Tom Hirschowitz, and Thomas Seiller. An intensionally fully-abstract sheaf model for π. *CALCO*, 2015.
14. Rob J. van Glabbeek and Gordon D. Plotkin. Configuration structures, event structures and petri nets. *Theor. Comput. Sci.*, 410(41):4111–4159, 2009.
15. Jonathan Hayman and Glynn Winskel. Event structure semantics for security protocols. Submitted for publication, 2013.
16. Jean Krivine. A verification algorithm for declarative concurrent programming. Technical Report 0606095, INRIA-Rocquencourt, 2006.
17. Robin Milner. *Communication and Concurrency*. Prentice-Hall, Inc., 1989.
18. Robin Milner. *Communicating and Mobile Systems: The π-calculus*. Cambridge University Press, 1999.
19. Iain Phillips and Irek Ulidowski. Reversibility and models for concurrency. *Electron. Notes Theor. Comput. Sci.*, 192(1):93–108, October 2007.
20. Davide Sangiorgi and David Walker. *PI-Calculus: A Theory of Mobile Processes*. Cambridge University Press, New York, NY, USA, 2001.
21. Vladimiro Sassone, Mogens Nielsen, and Glynn Winskel. Models for concurrency: towards a classification. *Theor. Comput. Sci.*, 170(1–2):297 – 348, 1996.
22. Glynn Winskel. Event structure semantics for ccs and related languages. In *Proceedings of the 9th Colloquium on Automata, Languages and Programming*, pages 561–576, 1982.
23. Glynn Winskel. Event structures. In *Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986, Part II*, volume 255 of *LNCS*, pages 325–392, 1986.
24. Nobuko Yoshida. Minimality and separation results on asynchronous mobile processes. *Theor. Comput. Sci.*, 274(1–2):231 – 276, 2002. Ninth International Conference on Concurrency Theory 1998.

A Appendix

A.1 Alternative definition of rigid families [6,15]

Let $\mathcal{F}$ a family of partial orders and $X \subseteq \mathcal{F}$. X is *compatible* denoted $X \uparrow^{\text{fin.}}$ if $\exists y \in \mathcal{F}$ finite such that $\forall x \in X,\ x \preceq y$.

Definition 21 (Morphisms from [6,15]). *A morphism on rigid families $\sigma : \mathcal{F} \to \mathcal{F}'$ is a partial function on partial orders such that*

- *it preserves compatible joins:*

$$\sigma(\emptyset) = \emptyset \text{ and if } x \uparrow x' \text{ then } \sigma(x \cup x') = \sigma(x) \cup \sigma(x')$$

- *it reflects contraction:*

$$\forall x \in \mathcal{F},\ \text{if } \sigma(x) \neq \emptyset \text{ then } \exists z \in \mathcal{F} \text{ such that}$$
$$\sigma(z) \prec \sigma(x) \text{ and } \nexists y \in \mathcal{F}', \sigma(z) \prec y \prec \sigma(x)$$

- *is partial injective on compatible configurations:*

$$\sigma(x) = \sigma(y) \text{ and } x \uparrow y \text{ then } \sigma(x) = \sigma(y) = \sigma(x \cap y)$$

There are two differences with the morphisms in definition 2:

1. Morphisms reflect (but do not necesarily preserve) the order in the configurations. Such morphisms induce a different product than the one defined in definition 1 where not all the partial orders we need are generated. We can change the first two conditions to the following (stronger) one:

$$\forall x, z \in \mathcal{F}, \text{ such that } \sigma(x), \sigma(z) \text{ defined } \sigma(z) \preceq \sigma(x) \iff z \preceq x \qquad (3)$$

   which requires of morphisms to preserve the order as well.
2. Partial injectivity of Definition 21 does not coincide with local injectivity Definition 2: it acts differently when events can synchronise. Consider the example in Figure 6. We cannot eliminate configuration $\{\tau < a\}$ with partial injectivity. In [15] events do not synchronize hence the partial injectivity is sufficient.

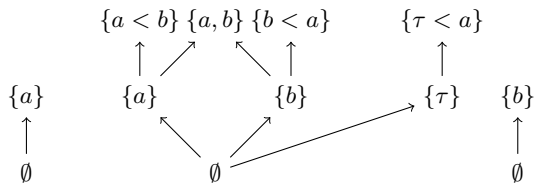

Fig. 6. Product

A.2 Rigid families and event structures

Definition 22 (Prime event structures).
A prime event structure $(E, \leq, \#)$ *consists of:*

- *a set of events* E*;*
- *a partial order on events* $\leq$*, called* causality *such that* $\forall e \in E$*, the set* $\{e' \in E \mid e' \leq e\}$ *is finite;*
- *and a binary irreflexive relation on events, called* conflict*, that is* hereditary*:*

$$\forall e_1, e_2, e_3 \in E, e_1 \# e_2 \leq e_3 \implies e_1 \# e_3.$$

A configuration *in a prime event structure is a set of events downward closed and conflict free:*

$$x \subseteq E, \forall e, e' \in E, \neg(e \# e') \text{ and } \forall e \in x, \forall e' \in E, e' \in x.$$

Denote $\mathcal{D}(E)$ *the set of configurations of* E*.*

Definition 23 (Order-reflecting morphisms). *A morphism on prime event structures* $f : (E_1, \leq_1, \#_1) \to (E_2, \leq_2, \#_2)$ *is a partial function on the underlying sets* $f : E_1 \to E_2$ *that is*

- configurations preserving: $\forall x \in \mathcal{D}(E_1), f(x) = \{f(e) \mid e \in x\} \in \mathcal{D}(E_2)$
- local injective: $\forall x \in C_1, \forall e_1, e_2 \in x, f(e_1) = f(e_2) \implies e_1 = e_2$

We can obtain a rigid family from a prime event structure.

Proposition 6 (Rigid family of an event structure). *Define the family of configuration of an event structure as all the conflict free, downward closed subsets of* E *ordered by inclusion. The order in each configuration* x *is the restriction of causal dependency* $\leq$ *to* $|x|$*. The partial orders of a prime event structure* $\mathcal{E} = (E, \leq, \#)$ *forms a rigid family* $\mathcal{D}(\mathcal{E}) = (E, C)$*. The construction* $\mathcal{D} : (E, \leq, \#) \to (E, \mathcal{L}(E)$ *is a functor.*

Proof. From [6].

Conversely, we can obtain an event structure $\mathcal{P}(\mathcal{F})$ from a rigid family by considering its primes.

Definition 24 (Complete primes). *A prime of* $\mathcal{F}$ *is a partial order* $[e]_x$ *for* $e \in x \in \mathcal{F}$*, where* $[e]_x$ *denotes the restriction of the partial order* x *to elements less than or equal to* e*:*

$$[e]_x = \{u < u' | u <_x e \ \& \ u <_x u'\}.$$

Proposition 7 (Event structures of a rigid family). *Let* (E, C) *a rigid family. The triple* $(P, \leq |_P, \#)$ *is a prime event structure, where*

- P *is the set of complete primes in* C*;*
- $\leq |_P$ *is the rigid inclusion of* C *restricted to the elements in* P*;*

– $p \# p' \iff \nexists x \in D$ *such that* $p \leq x$ *and* $p' \leq x$.

The map $\mathcal{P} : (E, C) \to (P, \leq |_P, \#)$ *is a functor between the categories of rigid families and the prime event structures.*

Proof. From [6].

If morphisms are defined as in Definition 21 then the functors $\mathcal{P}$ and $\mathcal{D}$ yield an equivalence of categories. However if morphisms preserve causality as in Equation 3, then it not longer holds. Consider the morphism between $\mathcal{C}_1$ and $\mathcal{C}_2$ in figure 7. As it does not preserve the order it has no correspondence for $\mathcal{D}(\mathcal{C}_1)$ and $\mathcal{D}(\mathcal{C}_2)$.

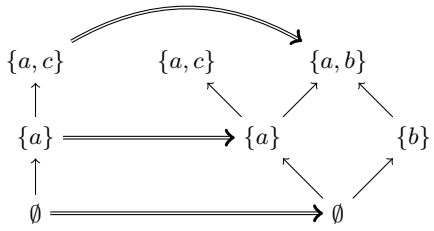

Fig. 7. Morphism in prime event structures

Another possible functor from rigid families to configuration structures is a *forgetful* one: we map partial orders to their underlying set. Let us denote $\mathcal{U} : \mathcal{F} \to \mathcal{C}$. We have that $\mathcal{U}$ is a right adjoint with $\mathcal{D}$ the left adjoint. However the construction is not interesting as we loose all causal informations.

Let us now consider $\mathcal{E}$ to be the category of event structures with *rigid morphisms*.

Definition 25 (Order-preserving and reflecting morphisms). *A rigid morphism on prime event structures* $f : (E_1, \leq_1, \#_1) \to (E_2, \leq_2, \#_2)$ *is a morphism (Definition 23) that preserves the order:*

$$e \leq e' \implies f(e) \leq f(e'), \text{ for } e, e' \in E_1 \text{ and } f(e), f(e') \text{ defined.}$$

The construction $\mathcal{D}$ is a left adjoint and $\mathcal{P}$ the right adjoint between the two categories with rigid morphisms.

Proposition 8. *Let* $\mathcal{E}$ *an object in the category of event structures and rigid map (Definition 25) and let* $\mathcal{F}$ *in the category of rigid families and rigid maps (Definition 2). Then*

1. $\mathcal{D} : \mathcal{E} \to \mathcal{F}$ *(Proposition 6) and* $\mathcal{P} : \mathcal{F} \to \mathcal{E}$ *(Proposition 7) are functors;*
2. $\mathcal{D} : \mathcal{E} \to \mathcal{F}$ *is the left adjoint to* $\mathcal{P} : \mathcal{F} \to \mathcal{E}$.

Proof. 1. Let us show $\mathcal{D} : \mathcal{E} \to \mathcal{F}$ is a functor. We have that $\mathcal{D}(\mathcal{E})$ is an event structure from [6]. We have to show that $\mathcal{D}$ preserves the identity and the composition laws on morphisms. Let $g : \mathcal{E}_1 \to \mathcal{E}_2$ and $\mathcal{D}(g) : \mathcal{D}(\mathcal{E}_1) \to \mathcal{D}(\mathcal{E}_2)$ defined as follows:

$$\mathcal{D}(g)(e) = g(e) \text{ and } \mathcal{D}(g)(x) = x' \iff g(|x|) = |x'|. \tag{4}$$

As g preserves and reflects the order and is local injective, $\mathcal{D}(g)$ is a rigid morphism on rigid families (Definition 2). The identity morphisms is preserved

$$\mathcal{D}(\mathsf{id}_{\mathcal{E}}) = \mathsf{id}_{\mathcal{D}(\mathcal{E})}$$

which follows from the definition of $\mathcal{D}(g)$. The composition law is satisfied due to the composition of partial functions on the underlying set of events. Similarly for $\mathcal{P} : \mathcal{F} \to \mathcal{E}$, we have that $\mathcal{P}(\mathcal{F})$ is a rigid family from [6]. Let $f : \mathcal{F}_1 \to \mathcal{F}_2$ and $\mathcal{P}(f) : \mathcal{P}(rfam_1) \to \mathcal{P}(\mathcal{F}_2)$ defined below:

$$\mathcal{P}(f)(p_1) = p_2 \iff f(p_1) = p_2.$$

The rest is similar to above.

2. We prove this by showing the following natural isomorphism:

$$\frac{\mathcal{D}(\mathcal{E}) \to \mathcal{F}}{\mathcal{E} \to \mathcal{P}(\mathcal{F})}$$

that is we show that $\forall \mathcal{E}$ an event structure and $\forall \mathcal{F}$ a rigid family we can define a family of bijections between $\hom(\mathcal{D}(\mathcal{E}), \mathcal{F})$ and $\hom(\mathcal{E}, \mathcal{P}(\mathcal{F}))$. We denote $\mathcal{E} = (E, \leq, \#)$, $\mathcal{F} = (E', C, \preceq)$ and $\mathcal{P}(\mathcal{F}) = (P, \leq_P, \#_P)$. Whenever we write $f(e)$ or $g(e)$, for $e \in E \cup F \cup P$, we assume that the function is defined on the event.

- Let us first define the morphism between $\hom(\mathcal{D}(\mathcal{E}), \mathcal{F})$ and $\hom(\mathcal{E}, \mathcal{P}(\mathcal{F}))$. Let $f : \mathcal{D}(\mathcal{E}) \to \mathcal{F}$. We have that, in $\mathcal{D}(\mathcal{E}) = (E, \mathcal{D}(E))$ the causal set of an event is unique:

  $$\forall e \in E, \forall x, x' \in \mathcal{D}(E) \text{ s.t. } e \in x, e \in x' \text{ we have that } [e]_x = [e]_{x'}.$$

  Hence we can write $[e]$ for the causal set of e in $\mathcal{D}(\mathcal{E})$. Define $g : \mathcal{E} \to \mathcal{P}(\mathcal{F})$ as follows:

  $$g(e) = p \iff f([e]) = p, \text{ where } p \text{ is a complete prime in } \mathcal{F}. \tag{5}$$

  Let us show that g is a morphism as defined by Definition 25.

  - g preserves configurations: $\forall x \in \mathcal{D}(E)$, $g(x) = \{g(e) \mid e \in x\} \in C$. We have that for $x \in \mathcal{D}(E)$, $x = \cup_{e \in x}[e]_x$. Let us develop $g(x)$:

    $$g(x) = \{p \in P \mid f([e]) = p, e \in x\} = \cup_{e \in x} f([e]) = \cup_{p \subseteq f(x)} p.$$

    As the set $y = \{p \in P \mid \forall p \subseteq f(x)\}$ is conflict free, we have that $y \in \mathcal{DP}(\mathcal{F})$.

- g is local injective: $\forall x \in \mathcal{D}(E), \forall e_1, e_2 \in x, g(e_1) = g(e_2) \implies e_1 = e_2$.
  We have that $g(e_1) = g(e_2)$ implies $f([e_1]_x) = f([e_2]_x)$, hence $e_1 = e_2$.
- g preserves the order: $e_1 \leq e_2 \implies g(e_1) \leq g(e_2)$, for $e_1, e_2 \in E$.
  Follows from f preserving the order: $e_1 \leq e_2 \implies [e_1] \subseteq [e_2] \implies f([e_1]) \preceq f([e_2])$.

– Let $g : \mathcal{E} \to \mathcal{P}(\mathcal{F})$. We define the additional map $h : \mathcal{D}(\mathcal{P}(\mathcal{F})) \to \mathcal{F}$ from which we get $f = \mathcal{D}(g) \circ h$:

$$f : \mathcal{D}(\mathcal{E}) \xrightarrow{\mathcal{D}(g)} \mathcal{D}(\mathcal{P}(\mathcal{F})) \xrightarrow{h} \mathcal{F}. \tag{6}$$

Define $h : \mathcal{D}(\mathcal{P}(\mathcal{F})) \to \mathcal{F}$ as follows:

$$h(p) = e' \iff g(e) = p \text{ where } p \in P \text{ and } p = [e']_p. \tag{7}$$

We can write $p = [e']_p$ as p is a complete prime. Let us show that h is a morphism as in Definition 2.

- h is local injective: $\forall x \in \mathcal{D}(E), e_1, e_2 \in x, h(e_1) = h(e_2) \implies e_1 = e_2$.
  If $h(e_1) = h(e_2)$ then $g(e_1) = g(e_2)$, and as g is local injective, we have that $e_1 = e_2$.
- h extends to a (total) function on configurations such that the causal order on events is preserved and reflected. For all $x \in \mathcal{D}(P)$, $x = \cup_{p \subseteq x} p$ and $\forall p_1, p_2 \in x$, $p_1 \leq_x p_2 \iff p_1 \leq_P p_2 \iff p_1 \preceq p_2$. Let $p_1 = [e'_1]_{p_1}$ and $p_2 = [e'_2]_{p_2}$. We can show that if $p_1 \preceq p_2$ then $\forall x' \in C$ such that $e'_1, e'_2 \in x'$ we have that $e'_1 \leq_{x'} e'_2$.
  It implies then that the set $\cup_{p \subseteq x} p$ is conflict free and downward closed and has the partial order $p_1 \leq p_2$.

To show that the constructions above define a bijection we only have to check that the bijection works on the underlying partial functions on events. Due to the rigid morphisms, the configurations are in a bijection if the underlying sets of events are. Therefore we show that $f(e) = f'(e)$ where

$$\mathcal{D}(\mathcal{E}) \xrightarrow{f} \mathcal{F} \Longrightarrow \mathcal{E} \xrightarrow{g} \mathcal{P}(\mathcal{F}) \Longrightarrow \mathcal{D}(\mathcal{E}) \xrightarrow{f'} \mathcal{F}.$$

First note that $f(e) = e'$, where $e \in E$ and $e' \in F$. Then we can derive that $f([e]) = [e']_x$ for some $x \in \mathcal{F}$. We denote $[e']_x = p$ a complete prime of $\mathcal{F}$. From Equation 5, we have that $g(e) = p \iff f([e]) = p$.
Let us now derive $f'(e)$. From Equation 6 we have that $f'(e) = h\big(\mathcal{D}(g)(e)\big)$. We use Equation 4 and obtain $f'(e) = h\big(g(e)\big)$, which from Equation 7, becomes $f'(e) = e' \iff g(e) = p$ and $p = [e']_p$.
Let us show that $g(e) = g'(e)$ for

$$\mathcal{E} \xrightarrow{g} \mathcal{P}(\mathcal{F}) \Longrightarrow \mathcal{D}(\mathcal{E}) \xrightarrow{f} \mathcal{F} \Longrightarrow \mathcal{E} \xrightarrow{g'} \mathcal{P}(\mathcal{F}).$$

We obtain that $f(e) = h(g(e))$ similar to above. Hence $f(e) = e' \iff g(e) = p$ and $p = [e']_p$, for $p \in \mathcal{P}(\mathcal{F})$, $e' \in F$ (Equation 7). From Equation 5, we have that $g'(e) = p' \iff f([e]) = p'$. But $f(e) = e'$, $f([e]) = [e']_{p'}$ and $p = [e']_p$. Hence $p' = p$.

But even if both categories use the rigid morphisms, there is no equivalence between the event structures and the rigid families. One cannot define an isomorphism between $\mathcal{D}(\mathcal{P}(\mathcal{F}))$ and $\mathcal{F}$ but only the co-unit $\mathcal{D}(\mathcal{P}(\mathcal{F})) \to \mathcal{F}$. To see this, consider the example of Figure 8. There is no morphism between $\mathcal{F}_1$ and $\mathcal{F}_2$ that preserves configurations.

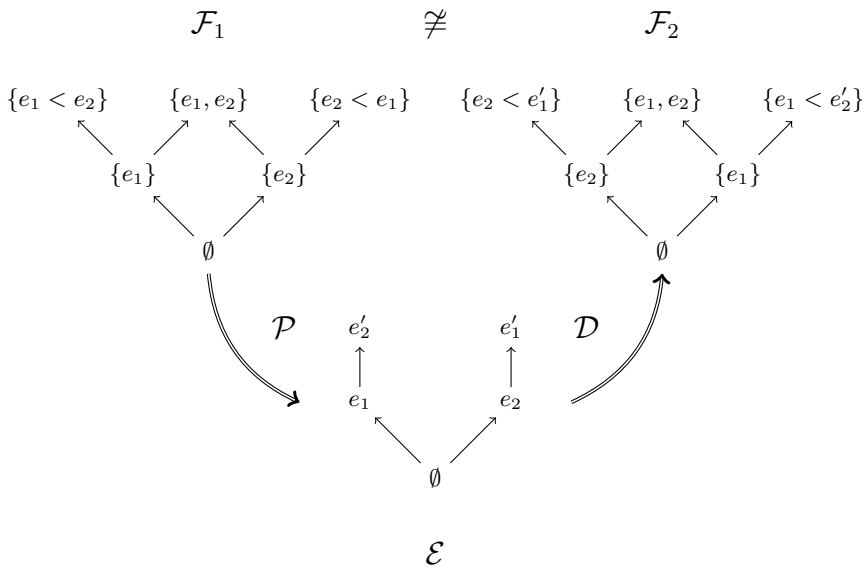

Fig. 8. $\mathcal{E} = \mathcal{P}(\mathcal{F}_1)$ and $\mathcal{F}_2 = \mathcal{D}(\mathcal{P}(\mathcal{F}_1))$

A.3 Proofs of section 2

Proof (Proposition 1). The category of rigid families and rigid morphisms consists in:

- a collection of rigid families: $\mathcal{F}_1, \mathcal{F}_2 \ldots$.
- a collection of rigid morphisms: $\sigma : \mathcal{F} \to \mathcal{F}$.
- for every pair of morphisms $\sigma : (E, C) \to (E', C')$ and $\sigma : (E', C') \to (E'', C'')$ a composite morphism $f \circ g : (E, C) \to (E'', C'')$. We have that $\sigma : E \rightharpoonup E'$ and let $D \subseteq E$ such that $\sigma : D \to E'$ is a total function. Similarly for $D' \subseteq E''$ and $\sigma' : D' \to E''$ total. Then $\sigma \circ \sigma' : D \to D' \to E''$

is a composition of total functions and $\sigma \circ \sigma' : E \supseteq D \to E' \supseteq D' \to E''$ is partial function undefined on $\{e \mid e \in E/D' \text{ or } e \in D, \sigma(e) \in E'/D'\}$. We have that $f \circ g$ preserves and reflects the order and is local injective from σ and σ' morphisms.
- for every rigid family $\mathcal{F}$ an identity morphism $\mathrm{id}_{\mathcal{F}} : \mathcal{F} \to \mathcal{F}$.
- the morphisms satisfy the following properties:
  - composition is associative;
  - identity is the unit for composition.

Proof (Equation 1). Denote $\mathcal{F}_1 = (E_1, C_1)$, $\mathcal{F}_2 = (E_2, C_2)$ and $\mathcal{F} = (E, C) = \mathcal{F}_1 \times \mathcal{F}_2$. Also we denote with π_i the projections on rigid families and σ_i the projections on sets, conform Equation 1. We show the following:

1. $\mathcal{F}$ is a rigid family;
2. π_1 and π_2 are morphisms: this is imposed by the definition;
3. $\mathcal{F}$ has the universal property of the product.

For the first condition suffices to show that any configuration is downward closed. We proceed by induction on the size of the configurations. Let $x \in C$ and let $y \preceq x$. We show that $y \in C$ by showing that the following holds:

$$\begin{aligned} &y \text{ is a partial order} && (\text{from } y \preceq x)\\ &\pi_1(y) \preceq \pi_1(x) && (\text{from } y \preceq x \text{ and the rigid morphisms})\\ &\pi_1(y) \in C_1 && (\text{as } \pi_1(x) \in C_1 \text{ and downward closed}) \end{aligned}$$

Let us now show the third condition: $\mathcal{F}$ has the universal property, as described by the diagram Figure 9. Let $\mathcal{F}' = (E', C')$ with $\pi'_1 : (E', C') \to (E_1, C_1)$ and $\pi'_2 : (E', C') \to (E_2, C_2)$ morphisms. We first observe that events in E are

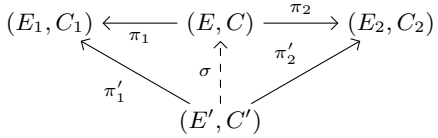

Fig. 9. Universal property of the product

uniquely identified by their projections:

$$\forall e, e' \in E, \sigma_1(e) = \sigma_1(e') \text{ and } \sigma_2(e) = \sigma_2(e') \implies e = e'. \tag{8}$$

Moreover note that $\nexists e \in E$ such that $\sigma_1(e) = \sigma_2(e) = \star$.

In order for the diagram to commute the mapping $\sigma : (E', C') \to (E, C)$ has to respect the projections:

$$\begin{aligned} \sigma(e') = e \iff & && \pi_1(e) = \pi'_1(e'), \text{ if } \pi'_1(e') \text{ defined}\\ & && \pi_2(e) = \pi'_2(e'), \text{ if } \pi'_2(e') \text{ defined}\\ = \text{ undefined} & && \text{otherwise} \end{aligned}$$

and extend σ to configurations

$$\sigma(x') = \{\sigma(e) \leq \sigma(e') \mid \sigma(e), \sigma(e') \text{ both defined and } e \leq_{x'} e'\}.$$

To show that σ is a morphism remains to show that σ is local injective: suppose $\exists e'_1, e'_2 \in x'$ such that $\sigma(e'_1) = \sigma(e'_2) = e_1$. It implies that $\pi'_1(e'_1) = \pi'_1(e'_2)$ and $\pi'_2(e'_1) = \pi'_2(e'_2)$, hence either π'_1 or π'_2 are not local injective. Contradiction.

Lastly let us show that σ is the unique morphism $\sigma : (E', C') \to (E, C)$ such that:

$$\forall e' \in E', \sigma(e') = e \implies \pi_1(e) = \pi'_1(e') \text{ and } \pi_2(e) = \pi'_2(e')$$

which follows from Equation 8.

A.4 Proofs of section 3

We recall the LTS of CCS in Figure 10.

$$\frac{}{\alpha.P \xrightarrow{\alpha} P} \quad \frac{P \xrightarrow{\alpha} P'}{\nu a P \xrightarrow{\alpha} \nu a P'} \text{ if } a \notin \alpha \quad \frac{P \xrightarrow{\overline{a}} P' \quad Q \xrightarrow{a} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'} \quad \frac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q}$$

Fig. 10. LTS of CCS

Proof (Lemma 1). Let P a process and $[\![P]\!] = (E, C, \ell, \mathsf{P})$ its encoding.

1. $\forall \alpha$, P' such that $P \xrightarrow{\alpha} P'$, $\exists e \in E$ such that $\ell(e) = \alpha$ and $[\![P]\!]/e \cong [\![P']\!]$. We proceed by induction on the derivation of the rule $P \xrightarrow{\alpha} P'$.
2. $\forall e \in E$, $\{e\} \in C$, $\exists P'$ such that $P \xrightarrow{\ell(e)} P'$ and $[\![P]\!]/e \cong [\![P']\!]$. We proceed by structural induction on P.

Definition 26 (Height of an event). *Let (E, C, ℓ, P) a configuration structure. For $x \in C$ and $e \in x$ define*

$$h_x(e) = |y|, \text{ where } y \subseteq x, e \in y \text{ and } \nexists z \subset y, e \in z$$

Proof (Proposition 5). Let $x \in C$ and $e_1, e_2 \in x$ such that $e_1 \diamond_x e_2$. We show that there exists a configuration $y \in C$ satisfying the conditions:

1. $|x| = |y|$;
2. y preserves the order in x: $\forall e, e' \in x$, $e \leq_x e' \implies e \leq_y e'$;
3. x reflects the order in y except for $e_1 <_y e_2$: $\forall e, e' \in y$ such that $\neg(e = e_1 \wedge e' = e_2)$, $e \leq_y e' \implies e \leq_x e'$.

We proceed by induction on the structure of $[\![P]\!]$.

– $[\![P]\!] = \alpha.[\![P']\!]$. It follows by induction on $[\![P']\!]$.

– $[\![P]\!] = [\![P_1]\!] \times [\![P_2]\!]$. From $x \in [\![P]\!]$ we have that $\pi_1(x) \in [\![P_1]\!]$ and $\pi_2(x) \in [\![P_2]\!]$. We proceed by cases on e_1, e_2.
  - If $\pi_1(e_1) = \pi_1(e_2) = \star$ then $\pi_2(e_1) \diamond_{\pi_2(x)} \pi_2(e_2)$. By induction on $\pi_2(x) \in [\![P_2]\!]$ we have that there exists $y_2 \in [\![P_2]\!]$ such that
    * $|\pi_2(x)| = |y_2|$;
    * $\forall e, e' \in y_2$ such that $\neg(e = \pi_2(e_1) \wedge e' = \pi_2(e_2))$, $e \leq_{y_2} e' \implies e \leq_{\pi_2(x)} e'$ (the second and third condition).

    Then let $y \in [\![P_1]\!] \times [\![P_2]\!]$ such that
    * $|x| = |y|$;
    * $\pi_1(x) = \pi_1(y)$ and $\pi_2(x) = y_2$;
    * $\forall e, e' \in y$ such that $\neg(e = e_1 \wedge e' = e_2)$, $e \leq_y e' \iff e \leq_x e'$ which is well defined due to the conditions on y_2.

    which is a partial order with projections well defined. Moreover conditions (1) to (3) are verified.
  - If $\pi_1(e_1) = \star$ and $\pi_1(e_2) \neq \star$ then let y such that
    * $|x| = |y|$;
    * $\pi_1(x) = \pi_1(y)$ and $\pi_2(x) = \pi_2(y)$;
    * $\forall e, e' \in y$ such that $\neg(e = e_1 \wedge e' = e_2)$, $e \leq_y e' \iff e \leq_x e'$.

    which staisfies conditions above. Remains to show that such y is a configuration in $[\![P_1]\!] \times [\![P_2]\!]$. It follows from being a partial order with te projections well defined.
– $[\![P]\!] = [\![P']\!] \upharpoonright X$. If $x \in [\![P]\!]$ then $x \in [\![P']\!]$ and by induction, there exists $y \in [\![P']\!]$ such that properties (1)-(3) hold in $[\![P']\!]$. Remains to show that $y \notin X$, hence $y \in [\![P]\!]$ to conclude. We have that if $e \in y$ then $e \in x$ and moreover, $\ell(e)$ does not change from x to y. Then, from Definition 8, if $x \notin X$ it implies that $y \notin X$ neither.

A.5 Proofs of section 4

Let us revisit Definition 13 to have formally introduce substitution.

Definition 27 (Substitution(extended)). *We define σ_x by induction on x:*

$$
\begin{aligned}
\sigma_\emptyset =& \emptyset \\
\sigma_x =& \sigma_{x\setminus e} \text{ if } \ell(e') \neq (d(a), \overline{b}\langle a' \rangle) \\
& \sigma_{x\setminus e} \cup \{a'/a\} \text{ if } \ell(e') = (d(a), \overline{b}\langle a' \rangle) \text{ and } \{a''/a'\} \notin \sigma_{x\setminus e} \\
& \sigma_{x\setminus e} \cup \{a''/a\} \text{ if } \ell(e') = (d(a), \overline{b}\langle a' \rangle) \text{ and } \{a''/a'\} \in \sigma_{x\setminus e}
\end{aligned}
$$

Define $\ell_x(e) = \ell(e)\sigma_x$, where

$$
\begin{aligned}
\big(d(a)\big)\sigma_x =& d'(a) && \text{if } \{d'/d\} \in \sigma_x \\
& d(a) && \text{otherwise} \\
\big(\overline{b}\langle a \rangle\big)\sigma_x =& \overline{b'}\langle a' \rangle && \text{if } \{b'/b\}, \{a'/a\} \in \sigma_x \\
& \overline{b}\langle a \rangle && \text{otherwise} \\
\big((\alpha, \beta)\big)\sigma_x =& \big((\alpha)\sigma_x, (\beta)\sigma_x\big)
\end{aligned}
$$

Consider as an example the process $P = \nu b, c(b(a).P \mid c(a').\bar{b}\langle a'\rangle \mid \bar{c}\langle a''\rangle)$ and let x be the maximal configuration in the encoding of P. We obtain the set of substitutions $\sigma_x = \{a''/a, a''/a'\}$.

Proposition 9 (σ_x well defined). *Let (E, C, ℓ, P) a rigid family such that it respects the Barendregt convention and such that for $x \in C$, $e \in E$ and $e' \in \mathsf{inst}_x(e)$ we have that $e' <_x e$. Then σ_x is well defined.*

Proof. As we have a single instantiator for each name a used in x, there is at most one substitution on a. Moreover the instantiator of a name (i.e. e such that $b(a) \in \ell(e)$) always precedes an output of that name (i.e. for e' such that $\overline{b'}\langle a\rangle \in \ell(e')$ we have $e' < e$).

We recall the LTS of π-calculus in Figure 11.

$$\frac{}{b(c).P \xrightarrow{b(c)} P} \quad \frac{}{\bar{b}\langle a\rangle.P \xrightarrow{\bar{b}\langle a\rangle} P} \quad \frac{P \xrightarrow{\bar{b}\langle a\rangle} P'}{\nu a P \xrightarrow{\bar{b}\langle \nu a\rangle} P'} \quad \frac{P \xrightarrow{\alpha} P'}{\nu a P \xrightarrow{\alpha} \nu a P'} \text{ if } a \notin \alpha$$

$$\frac{P \xrightarrow{\bar{b}\langle a\rangle} P' \quad Q \xrightarrow{b(c)} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'\{a/c\}} \quad \frac{P \xrightarrow{\bar{b}\langle \nu a\rangle} P' \quad Q \xrightarrow{b(c)} Q'}{P \mid Q \xrightarrow{\tau} \nu a(P' \mid Q'\{a/c\})} \text{ if } a \notin \mathrm{fn}(Q)/\{c\}$$

$$\frac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q} \text{ if } \mathrm{bn}(\alpha) \cap \mathrm{fn}(Q) = \emptyset$$

Fig. 11. LTS of π calculus

Proof (Lemma 2). Let P a process and $[\![P]\!] = (E, C)$ its encoding.

- $\forall \alpha$, P' such that $P \xrightarrow{\alpha} P'$, $\exists e \in E$ such that $\ell_e(e) = \alpha$ and $[\![P]\!]/\{e\} \equiv [\![P']\!]$.
  We proceed by induction on the derivation of the rule $P \xrightarrow{\alpha} P'$.
  - $\alpha.P \xrightarrow{\alpha} P$.
    We have $[\![\alpha.P]\!] = \alpha.[\![P]\!]$ hence we have to show that

    $$[\![P]\!] \cong (\alpha.[\![P]\!])/\{e\}, \text{ where } \ell_{\{e\}}(e) = \alpha. \tag{9}$$

    If we want to be rigorous we need to define the isomorphism via the projections $\pi_\alpha : [\![\alpha.P]\!] \to [\![P]\!]$ and $\pi_e : ([\![\alpha.P]\!])/\{e\} \to [\![\alpha.P]\!]$. For the sake of simplicity we consider instead π_α and π_e to be the identity on all events in $[\![P]\!]$ and adopt this as a convention throughout the proofs. Then the isomorphism is the identity on events. Let us show that if $x_1 \in [\![P]\!]$ then $x_1 \in (\alpha.[\![P]\!])/\{e\}$ and have the other direction in a similar manner. If $x_1 \in [\![P]\!]$ then $x_1 \cup \{e\} \in [\![\alpha.P]\!]$ and $x_1 \in (\alpha.[\![P]\!])/\{e\}$.
  - $\frac{P \xrightarrow{\bar{b}\langle a\rangle} P' \quad Q \xrightarrow{b(c)} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'\{a/c\}}$.

Let P the set of private names in P and Q. We have $[\![P|Q]\!] = ([\![P]\!] \times [\![Q]\!]) \upharpoonright X$, where the private names are restricted by a subset of X. By induction $[\![P']\!] = [\![P]\!]/\{e_1\}$ where $\ell_{\{e_1\}}(e_1) = \overline{b}\langle a\rangle$ and $[\![Q']\!] = [\![Q]\!]/\{e_2\}$ where $\ell_{\{e_2\}}(e_2) = b(c)$. Then $\{e_1\} \in [\![P]\!]$ and $\{e_2\} \in [\![Q]\!]$ it implies that there exists $\{e_{12}\} \in [\![P]\!] \times [\![Q]\!]$ such that $\pi_1(e) = e_1$ and $\pi_2(e) = e_2$.
Hence we have to show that

$$\begin{aligned}\mathcal{F}_1 = \Big(\big([\![P]\!]/\{e_1\} \times [\![Q]\!]/\{e_2\}\big) \upharpoonright X'\Big)\{a/d\} \cong \\ \big(([\![P]\!] \times [\![Q]\!]) \upharpoonright X\big)/\{e_{12}\} = \mathcal{F}_2\end{aligned} \tag{10}$$

where $\mathcal{F}\{a/d\}$ replaces d with a in all labels.

Let $\mathcal{F}_1 = (E_1, C_1)$ and $\mathcal{F}_2 = (E_2, C_2)$. Denote $\pi_{1,P} : \mathcal{F}_1 \to [\![P]\!]$ and $\pi_{1,Q} : \mathcal{F}_1 \to [\![Q]\!]$ and similarly for $\pi_{2,P}$, $\pi_{2,Q}$. We obtain these moprhisms similarly to Definition 20. Consider the following function on event:

$$\iota(e_1) = e_2 \iff \pi_{1,P}(e_1) = \pi_{2,P}(e_2) \text{ and } \pi_{1,Q}(e_1) = \pi_{2,Q}(e_2).$$

To show it is an isomorphism we show that $x_1 \in \mathcal{F}_1 \iff x_2 \in \mathcal{F}_2$ and $\cong(x_1) = x_2$ preserving and reflecting the partial order.

$x_1 \in \mathcal{F}_1 \implies \iota(x_1) \in \mathcal{F}_2$**.** We proceed by induction on $<_{x_1}$. Trivial for the base case. For the inductive step, consider $x_1 = x_1' \cup \{e\}$ and $\iota(x_1') = x_2'$. We have that

$$\pi_{1,P}(x_1) = \pi_{1,P}(x_1') \cup \{\pi_{1,P}(e), e_1\}$$

and

$$\pi_{1,Q}(x_1) = \pi_{1,Q}(x_1') \cup \{\pi_{1,Q}(e), e_2\}.$$

Let $e' \in E_2$ such that $\pi_{1,P}(e) = \pi_{2,P}(e')$ and $\pi_{1,Q}(e) = \pi_{2,Q}(e')$. Let $x_2 = x_2' \cup \{e', e_{12}\}$ where $e_2' <_{x_2} e_2'' \iff \iota(e_2') <_{x_1} \iota(e_2'')$. Then x_2 is a partial order and is downward closed in $[\![P]\!] \times [\![Q]\!]$.
$x_2 \notin X'$ follows from $x_1 \notin X$, hence $x_2 \in [\![P]\!] \times [\![Q]\!]$. As $e_{12} \in x_2$ we have that $x_2 \setminus e_{12} \in \big(([\![P]\!] \times [\![Q]\!]) \upharpoonright X'\big)/\{e_{12}\}$.

$x_2 \in \mathcal{F}_2 \implies \iota(x_2) \in \mathcal{F}_1$**.** Similarly, we proceed by induction on $<_{x_2}$. Let $x_2 = x_2' \cup \{e\}$ and $\iota(x_2') = x_1'$. We have that

$$x_2 \in \big(([\![P]\!] \times [\![Q]\!]) \upharpoonright X'\big)/\{e_{12}\} \implies x_2 \cup \{e_{12}\} \in [\![[\![P]\!] \times [\![Q]\!]]\!]$$

with $x_2 \notin X'$. The rest of the reasoning follows the case above.

- $\dfrac{P \xrightarrow{\overline{b}\langle \nu a\rangle} P' \quad Q \xrightarrow{b(c)} Q'}{P \mid Q \xrightarrow{\tau} \nu a(P' \mid Q'\{a/c\})}$ if $a \notin \mathrm{fn}(Q)/\{c\}$. We use the same notations as in the case above. Hence we have to show that

$$\Big(\big([\![P]\!]/\{e_1\} \times [\![Q]\!]/\{e_2\}\big) \upharpoonright X'\Big)\{a/d\} \upharpoonright a \cong \big(([\![P]\!] \times [\![Q]\!]) \upharpoonright X\big)/\{e_{12}\}$$

But $X_a \subseteq X'$, hence it follows from equation 10.

- $\frac{P \xrightarrow{\alpha} P'}{\nu a P \xrightarrow{\alpha} \nu a P'}$, for $a \notin \alpha$. By induction we have that $[\![P]\!]/\{e\} = [\![P']\!]$, where $\ell_{\{e\}}(e) = \alpha$. We have then to show that $\{e\} \in [\![\nu a P]\!]$ and

$$([\![P]\!]/\{e\}) \upharpoonright a \cong ([\![P']\!] \upharpoonright a)/\{e\} \tag{11}$$

  Let $x \in ([\![P]\!]/\{e\}) \upharpoonright a$. Then $x \notin X_a$ and $x' \in [\![P]\!]$ such that $|x'| = |x| \cup \{e\}$. From $x \notin X_a$ and $a \notin \alpha$, $x' \in [\![P]\!] \upharpoonright a$, hence $x' \setminus \{e\} \in [\![P]\!]$. We have that $x' \setminus \{e\} = x$.
  For the other direction, consider $x \in ([\![P']\!] \upharpoonright a)/\{e\}$. It implies $x' \in [\![P]\!] \upharpoonright a$, $|x'| = |x| \cup \{e\}$. We have that $x' \notin X_a$, $x' \in [\![P]\!]$ and $x' \setminus \{e\} = x \in [\![P]\!]/\{e\}$. As $a \notin \alpha$ and $x' \notin X_a$ we conclude $x \in ([\![P]\!]/\{e\}) \upharpoonright a$.
- $\frac{P \xrightarrow{\overline{b}\langle a \rangle} P'}{\nu a P \xrightarrow{\overline{b}\langle \nu a \rangle} P'}$. By induction we have that $[\![P]\!]/\{e\} = [\![P']\!]$, where $\ell_{\{e\}}(e) = \overline{b}\langle \nu a \rangle$. We have then to show that $\{e\} \in [\![\nu a P]\!]$ and

$$([\![P]\!] \upharpoonright a)/\{e\} \cong [\![P]\!]/\{e\}, \text{ where } \ell_{\{e\}}(e) = \overline{b}\langle \nu a \rangle. \tag{12}$$

  From Equation 2 we have that $\{e\} \in [\![\nu a P]\!]$. Trivial to show that for $x \in ([\![P]\!] \upharpoonright a)/\{e\}$, $x \in [\![P]\!]/\{e\}$.
  Let us consider then $x \in [\![P]\!]/\{e\}$ and suppose $\exists e' \in x$ such that $a \in \mathsf{subj}_x(e')$. Then $x \cup \{e\} \in [\![P]\!]$.
  We have that there exists $x', x'' \in [\![P]\!]$ such that $|x'| = |x''| = |x| \cup \{e\}$ but $e <_{x'} e'$ and e, e' are incomparable in x''. We have that $x'' \in X_a$, $x' \notin X_a$. However $x' \setminus \{e\} = x'' \setminus \{e\}$, and $x' \setminus \{e\} \in ([\![P]\!] \upharpoonright a)/\{e\}$, hence $x \in ([\![P]\!] \upharpoonright a)/\{e\}$.

– $\forall e \in E$, $\{e\}$ complete $\exists P'$ such that $P \xrightarrow{\ell_e(e)} P'$ and $(E, C)/\{e\} \equiv [\![P']\!]$. We proceed by structural induction on P:
  - $P = \alpha.P'$ then $[\![P]\!] = [\![\alpha.P']\!] = \alpha.[\![P']\!]$. There exists only one singleton configuration $\{e\} \in \alpha.[\![P']\!]$, where $\ell_{\{e\}}(e) = \alpha$. We have that $\alpha.P' \xrightarrow{\alpha} P'$ hence $[\![P']\!] \cong (\alpha.[\![P]\!])/\{e\}$, which follows from equation 9.
  - $P = P_1 \mid P_2$ then $[\![P_1 \mid P_2]\!] = ([\![P_1]\!] \times [\![P_2]\!]) \upharpoonright X \upharpoonright X_{\mathsf{P}}$ with $\pi_1 : [\![P_1 \mid P_2]\!] \to [\![P_1]\!]$, $\pi_1 : [\![P_1 \mid P_2]\!] \to [\![P_1]\!]$ the two projections. Let $\{e\} \in [\![P_1 \mid P_2]\!]$ with $\pi_1(e) = e_1$, $\pi_2(e) = e_2$. The projections preserve configurations hence if $e_1 \neq \star$ then $\{e_1\} \in [\![P_1]\!]$, and similar for $\{e_2\} \in [\![P_2]\!]$.
    We only consider the case where $e_1 \neq \star$ and $e_2 \neq \star$. If $e_1 = \star$ or $e_2 = \star$ has a similar proof. Let $\ell_{\{e_1\}}(e_1) = \alpha_1$ and $\ell_{\{e_2\}}(e_2) = \alpha_2$. By the induction hypothesis, $P_1 \xrightarrow{\alpha_1} P_1'$ and $P_2 \xrightarrow{\alpha_2} P_2'$, where $[\![P_1']\!] = [\![P_1]\!]/\{e_1\}$ and $[\![P_2']\!] = [\![P_2]\!]/\{e_2\}$.
    Supose $\alpha_2 = c(d)$. We have the two following cases:
    * $\alpha_1 = \overline{b}\langle a \rangle$. Then $P_1|P_2 \xrightarrow{\tau} P_1'|P_2'$ and we have to show

$$\big([\![P_1]\!]/\{e_1\} \times [\![P_2]\!]/\{e_2\}\big) \upharpoonright X' \upharpoonright X_{\mathsf{P}} \cong \big(([\![P_1]\!] \times [\![P_2]\!]) \upharpoonright X \upharpoonright X_{\mathsf{P}}\big)/\{e\}$$

      which follows from equation 10.
    * $\alpha_1 = \overline{b}\langle \nu a \rangle$. Then $P_1|P_2 \xrightarrow{\tau} \nu a(P_1'|P_2')$ and we show the same as above.

- $P = \nu a(P')$ then $[\![P]\!] = [\![P']\!] \upharpoonright a$. We have that $\forall\{e\}[\![P]\!]$, $a \notin \mathsf{subj}_{\{e\}}(e)$ and $\{e\} \in [\![P]\!]$. By the induction hypothesis $P' \xrightarrow{\ell_{\{e\}}(e)} P''$ such that $[\![P'']\!] = [\![P']\!]/\{e\}$. We have two cases:
  - $a \notin \ell_{\{e\}}(e)$ or $\ell_{\{e\}}(e) = \tau$. Then $\nu a(P') \xrightarrow{\ell_{\{e\}}(e)} \nu a(P'')$ and we have to show that $[\![P'']\!] \upharpoonright a \cong ([\![P']\!] \upharpoonright a)/\{e\}$ that is
  $$([\![P']\!]/\{e\}) \upharpoonright a \cong ([\![P'']\!] \upharpoonright a)/\{e\}$$
  which follows from equation 11.
  - $\ell_{\{e\}}(e) = \overline{b}\langle a\rangle$. Then $\nu a(P') \xrightarrow{\overline{b}\langle \nu a\rangle} P''$ and we have to show $[\![P'']\!] \cong ([\![P']\!] \upharpoonright a)/\{e\}$ which follows from
  $$([\![P']\!] \upharpoonright a)/\{e\} = [\![P']\!]/\{e\}$$
  in equation 12.

Proposition 10 (Instantiator is a precedessor). *Let $[\![P]\!] = (E, C, \ell, \mathsf{P})$, $x \in C$ and $e \in x$. If $\exists e' \in x$, such that $e' = \mathsf{inst}_x(e)$ then $e' <_x e$.*

Proof. By induction on $[\![P]\!]$. The only interesting case if $P = b(a).P'$.

Proposition 11. *Let $[\![P]\!] = (E, C, \ell, \mathsf{P})$. $\forall x \in C$ and $\forall e_1, e_2 \in x$ such that $e_1 \diamond_x e_2$, $\exists y \in C$ such that $|x| = |y|$ and*

1. *y preserves the order in x: $\forall e, e' \in x$, $e \leq_x e' \implies e \leq_y e'$*
2. *x reflects the order in y except for $e_1 <_y e_2$: $\forall e, e' \in y$ such that $\neg(e = e_1 \wedge e' = e_2)$, $e \leq_y e' \implies e \leq_x e'$.*

Proof. By induction on $[\![P]\!]$. We follow the proof of Proposition 5. The only case that is different is for $[\![P]\!] = [\![P']\!] \upharpoonright X$. Let $x \in C$ and $e_1, e_2 \in x$ such that $e_1 \diamond_x e_2$. We have that $x \in [\![P']\!]$ and hence there exists y satisfying the hypothesis in $[\![P']\!]$. Remains to show that $y \notin X$.

From $|x| = |y|$ and Proposition 10 we have that $\forall e \in y$, $\ell_y(e) = \ell_x(e)$. We show by contradiction that if $e \in y$ does not satisfy conditions of Definition 15 and Definition 16 then $e \in x$ does not satisfy them either. Hence $x \in X$. Contradiction.

A.6 Proofs of section 5

Proposition 12. *Let $x \in [\![P]\!]$ and $e \in x$ such that $\ell_x(e) = (b(a), \overline{c}\langle a\rangle)$. Then at most one of b, c are private.*

Proof. By structural induction on $[\![P]\!]$. The only interesting case is when $P = \nu a.P'$ with $a \in \{b, c\}$. Suppose $a = b$. We have that $x \in [\![P]\!]$ then if for $e \in x$ such that $\ell_x(e) = (b(a), \overline{c}\langle a\rangle)$, there exists instantiator of c in x (second condition in Definition 15). Then we cannot add a restriction on c.

Definition 28 (Names in a configuration). *Define the set of names in a configuration x such that $a \in \text{names}(x) \iff \exists e \in x, a \in \ell_x(e)$.*

Proof (Theorem 3).
We proceed in several steps:

1. We show that for $x \in [\![P]\!]$ there exists a context $C_1 = \alpha_1 \cdots \alpha_n.[\cdot]$ and $x_1 \in [\![C_1[P]]\!]$ such that
   - $\pi_{C_1,P}(x_1) = x$ and
   - $\forall e \in x_1$, if $\tilde{\tau}_{x_1}(e)$ then $\forall b \in \mathsf{subj}(\ell_{x_1}(e))$ we have that

$$b \notin \mathsf{P} \implies \exists e_1 \in x_1, e_1 \in \mathsf{inst}^s_{x_1}(e) \text{ such that } \ell(e_1) = d'(b) \tag{13}$$

2. Define *precontext* which is a multiset of labels such that

$$b(a) \in \chi \implies c(a) \notin \chi, \text{ for any } c. \tag{14}$$

Let χ a precontext and let $f : \chi \to x_1$ a function. Define the set of substitutions implied by χ as follows:

$$\begin{aligned}
\varsigma(\emptyset) =& \emptyset \\
\varsigma(x \cup \{e\}) =& \{a'/a\} \cup \varsigma(x) \text{ if } \ell(e) = (d(a), \overline{b}\langle a' \rangle) \\
& \text{or if } \ell(e) = d(a), \alpha = \overline{b}\langle a' \rangle \text{ (or the converse) where } f(\alpha) = e \\
=& \{a''/a'\} \cup \big(\varsigma(x) \setminus \{a'/a\}\big) \text{ if } \ell(e) = (d(a'), \overline{b}\langle a'' \rangle), \\
& \text{or if } \ell(e) = d(a), f(e) = \overline{b}\langle a' \rangle \text{ (or the converse)}, \\
& \text{where } f(\alpha) = e, \{a'/a\} \in \varsigma(x)
\end{aligned}$$

For $x_1 \in [\![P_1]\!]$ we show that there exists a precontext χ and a function $f : \chi \to x_1$ such that the following holds:
(a) $\forall e \in x_1$ with $\ell_{x_1}(e) = (\overline{b}\langle a \rangle, c(d))$ then $\ell_\chi(e) = (\overline{b}\langle a \rangle, b(d))$;
(b) $\forall e \in x_1$ with $\ell_{x_1}(e) = \overline{b}\langle a \rangle$ or $\ell_{x_1}(e) = b(d)$ there exists $\alpha \in \chi$, $\alpha = b'(a')$ or $\alpha = \overline{b'}\langle a' \rangle$ respectively, such that $f(\alpha) = e$ and $b'\varsigma(x_1) = b$;
(c) if $b(a) \in \chi$ then $a \notin \text{Names}(x)$ and $\exists \alpha \in \chi$ such that $a = \mathsf{subj}(\alpha_2)$ iff $f(\alpha_1) <_{x_1} f(\alpha_2)$ and there is $c \in \ell_{x_1}(f(\alpha_2))$ private;
where $\ell_\chi(e) = \ell_{x_1}(e)\varsigma(x_1)$. Moreover f is total and injective: $\forall \alpha \in \chi$, $f(\alpha)$ defined and if $f(\alpha_1) = f(\alpha_2)$ then $\alpha_1 = \alpha_2$ (where $\alpha_1 = \alpha_2$ is not a syntactic equality).
We proceed by induction on the order $<_{x_1}$. We only treat the inductive case, as the base case is straightforward.

Let $x = x' \cup \{e\}$. From the induction hypothesis there exists χ' a precontext and $f : \chi' \to x'$ satisfying conditions above. We build the precontext χ and the function f such that $\chi = \chi'\sigma \cup \{\alpha\}$ and $f = f'\sigma \cup \{\alpha \mapsto e\}$, where σ is a list of substitutions. Both σ and α are defined below, by cases on $\ell_x(e)$:
- $\ell_\chi(e) = (\overline{b}\langle a \rangle, c(d))$. We proceed by cases on whether b and c are private:
  - $b, c \in \mathsf{P}$. From proposition 12 this case is not possible.

- $b \in \mathsf{P}$ and $c \notin \mathsf{P}$. Then from the hypothesis on x (Equation 13) we have that $c \notin \mathsf{P} \implies \exists e_c = \mathsf{inst}_x(c)$, where the instantiator of a name is defined similarly to the instantiator of an event.
  We have that $e_c \in x'$, then from the induction hypothesis on x' there exists $\alpha_c \in \chi$, $f(\alpha_c) = e_c$ and $\alpha_c = \overline{d}\langle c\rangle$.
  As e satisfy Definition 15 then there exists an extruder of b: let $e' \in x$, $e' <_x e$ and $\ell_x(e') = \overline{d'}\langle b\rangle$. Let $\alpha_b = d'(b')$ where $f(\alpha_b) = e'$.
  Then let $\chi = \chi'\{b'/c\}$ and $f = f'\{b'/c\}$. We have to show that:
  - f is a total and injective function, which follows from f'.
  - χ is a precontext: consists in showing that it does not exists an instantiator of c in χ'. Follows from $c \notin \mathsf{P}$.
  - the conditions above on χ hold. First note that $\ell_\chi(e) = (\overline{b}\langle a\rangle, b(d))$ as $b'\varsigma(x) = b \implies c\varsigma(x) = b$. Let $e' \in x$ such that $\ell_{\chi'}(e') = (\overline{c}\langle a'\rangle, c(d'))$. But then $\ell_\chi(e') = (\overline{b}\langle a'\rangle, b(d'))$. We have that α_b is an instantiator for α_c and that $b \in \mathsf{subj}(\ell_x(e))$ is private. Hence the thrid condition on the precontext holds as well.
- $b, c \notin \mathsf{P}$. Then there exists $e_c, e_b \in \mathsf{inst}^{\mathsf{s}}_x(e)$ (from Equation 13) and $e_c, e_b \in x'$. From χ' a precontext for x' we have that there exists $\alpha_b, \alpha_c \in \chi'$ such that $f(\alpha_b) = e_b$, $f(\alpha_c) = e_c$, and $\alpha_b = \overline{d}\langle b\rangle$, $\alpha_c = \overline{d}\langle c\rangle$.
  Then let $\chi = \chi'\{b/c\}$ and $f = f'\{b/c\}$. The first two conditions on the precontext hold from a similar reasoning as above. Note that there does not exists instantiators of b or c in χ as $b, c \notin \mathsf{P}$. Hence third condition holds as well.

– $\ell_\chi(e) = b(a)$. By cases on b:
- $b \in \mathsf{P}$. From the condition on the restriction operation in definition 16 we have that there exists an extruder $e' \in x$, $e' <_x e$ and $\ell_x(e') = \overline{c'}\langle b\rangle$. As $e' \in x'$, there exists $\alpha' \in \chi'$ such that $f(\alpha') = e'$ and hence $\alpha' = c'(d)$, with $d \notin \mathrm{names}(x)$. Moreover $d\varsigma(x) = b$.
  Define $\alpha = \overline{d}\langle a'\rangle$, a' fresh (i.e. $a' \notin \mathrm{names}(x)$) and $\chi = \chi' \cup \alpha$, $f = f' \cup \{\alpha \to e\}$.
  Let us show that χ verifies the conditions on the precontext and that f is total and injective. The later follows from $f(\alpha) = e$ and $e \notin x'$. The first condition on χ follows from χ' a precontext. The second condition is implied by $d\varsigma(x) = b$. The third condition is satisfied as well as for $\alpha' = \mathsf{inst}^{\mathsf{s}}(\alpha)$ we have that $b = \mathsf{subj}(e) \in \mathsf{P}$, $e' <_x e$.
- $b \notin \mathsf{P}$. Then let $\alpha = \overline{b}\langle a'\rangle$, with a' fresh and $\chi = \chi' \cup \alpha$, $f = f' \cup \{\alpha \to e\}$. The conditions on χ and f follow easily.

3. We construct a process P_2 from the precontext χ such that

$$\nexists a, \text{ such that } \nu a \in P_2 \tag{15}$$

$$\alpha \in P_2 \iff \alpha \in \chi \tag{16}$$

$$\alpha_1 \cdots \alpha_n.0 \in P_2 \iff \exists a \in \mathsf{obj}(\ell_{x_1}(e_1)), a \in \mathsf{P} \text{ and } e_n <_{x_1} e_i, a \in \mathsf{subj}(\ell_{x_1}(e_i)) \tag{17}$$

for $f(\alpha_1) = e_1$ and $f(\alpha_i) = e_i$, with $i \in \{2, n\}$. The context required by the theorem is $C = P_2 \mid [\cdot]$. Equation 15 guarantees that there are no private names in P_2, Equation 16 says that the context is build from the precontext and Equation 17 constructs a context that is maximal concurrent: the only sequential computation is due to the extrusion from P (and the instantiation in P_2) of a private name.
We proceed by induction on the precontext χ such that

$$\alpha_1 <_\chi \alpha_2 \text{ iff } f(\alpha_1) >_{x_1} f(\alpha_2).$$

The base case is trivial. Let us consider the inductive case. Let $\chi = \chi' \cup \{\alpha\}$ and $f = f' \cup \{\alpha \to e_1\}$. Let P_2' be the process generated by χ' satisfying equations above.
Let then the following set of events:

$$E_{e_1} = \{e_1' \in x_1 \mid e_1 <_{x_1} e_1', \nexists e'' \in x_1, e', e_1 <_{x_1} e'' <_{x_1} e_1' \text{ and} \tag{18}$$
$$\ell_{x_1}(e_1') \neq (\beta, \beta') \text{ and} \tag{19}$$
$$\ell_{x_1}(e_1) = \overline{b}\langle a \rangle, \text{ with } a \in \mathsf{P} \text{ and} \tag{20}$$
$$\exists e'', e_1 < e'' \text{ such that } a \in \mathsf{subj}(\alpha''), \text{ for } f(\alpha'') = e''\}. \tag{21}$$

We have that $\forall e' \in E_{e_1}$, $\exists \alpha' \in \chi'$ such that $f(\alpha') = e'$. This follows from Equation 18 and Equation 19. Then $\alpha' \in P_2'$. Equation 18 also guarantees that for $e' \in E_{e_1}$ with $f(\alpha') = e'$ we can rewrite $P_2 = \alpha'.Q' \mid R$. Equation 20 says that α is an instantiator. Moreover the name received is private. Lastly, Equation 21 guarantees that Equation 17 is satisfied for P_2 constructed below. We consider the following cases:

- If $E_{e_1} = \emptyset$ then $P_2 = \alpha \mid P_2'$ and Equation 15, Equation 16, Equation 17 trivially hold. P_2 is a valid π process as α is not an instantiator of any α' in P_2' (from third condition on the precontext and from Equation 14).
- Suppose then $E_{e_1} \neq \emptyset$. From Equation 17 on P_2', we can partition P_2' in the following way:

$$P_2' = \prod_{e' \in E_{e_1}} \alpha'.Q \mid R$$

where $f(\alpha') = e'$. We have then

$$P_2 = \alpha.(\prod_{e' \in E_{e_1}} \alpha'.Q) \mid R.$$

P_2 is a valid process if $b(a) \in \chi \implies c(a) \notin \chi$, for any c, which follows from the definition of a precontext in Equation 14. Equation 17 follows from the definition of E_e and the rest of the conditions follow trivially.

4. We show that $\exists x_2 \in [\![P_2]\!]$ and $g : x_2 \to x_1$ a total, injective function such that

$$\alpha \in P_2 \iff e \in x_2, \ell_{x_2}(e) = \alpha \tag{22}$$
$$g(e_2) = e_1 \iff f(\ell_{x_2}(e_2)) = e_1 \tag{23}$$
$$e_2 <_{x_2} e_2' \iff g(e_2) <_{x_1} g(e_1) \tag{24}$$

Inuitively, x_2 is the configuration that composes with x_1 in order to produce the closed configuration z. We have that x_2 is maximal in $[\![P_2]\!]$, hence it contains all labels in χ (Equation 22). Equation 23 and Equation 24 ensure that the function g keeps the correspondence between partial events in x_1 and their 'future' synchronisation partners in x_2.

Due to conditions Equation 23 and Equation 24 we cannot apply an inductive reasoning on P_2 to build x_2 directly. Instead we start by showing that there exists $y \in [\![P_2]\!]$ maximal in $[\![P_2]\!]$ (i.e. $\alpha \in P_2 \iff e \in y, \ell_y(e) = \alpha$) without any synchronisation event (i.e. $\forall e \in y,\ \ell_y(e) \neq (\alpha, \beta)$). Moreover y is the most liberal such configuration:

$$e <_y e' \iff \ell_y(e) \cdots \ell_y(e') \in P_2 \tag{25}$$

We proceed by structural induction on P_2.

- $Q = \beta.Q'$. By induction there exists $y' \in [\![Q']\!]$. We have that $[\![Q]\!] = [\![\beta.Q']\!] = \beta.[\![Q']\!]$ hence we have $y = \{e\} \cup y' \in [\![P_2]\!]$, where $\ell_y(e) = \beta$. Then y is maximal and satsifies Equation 25.
- $Q = Q_1 \mid Q_2$. By induction on Q_1 and Q_2 we have $y_1 \in [\![Q_1]\!]$ and $y_2 \in [\![Q_2]\!]$, respectively satsifying the required conditions. By definition of product, we have that $y \in [\![Q_1]\!] \times [\![Q_2]\!]$ if $\pi_1(y) \in [\![Q_1]\!]$ and $\pi_2(y) \in [\![Q_2]\!]$, where π_i are morphisms. Then we consider y such that
  - $\pi_1(y) = y_1$ and $\pi_2(y) = y_2$;
  - $e \in y$ iff either $\pi_1(e) \in y_1, \pi_2(e) = \star$ or $\pi_1(e) = \star, \pi_2(e) \in y_2$;
  - $e <_y e'$ iff either $\pi_1(e) <_{y_1} \pi_1(e')$, both defined or $\pi_2(e) <_{y_2} \pi_2(e')$, both defined.

  Then y is maximal (from the first item), contains no synchronisation (which follows from the second item) and satisfy Equation 25 (the thrid item).

Define function $g : y \to x_1$ total and injective as follows: $g(e_2) = e_1 \iff f(\ell_y(e_2)) = e_1$. We have $y \in [\![P_2]\!]$ with $|y| = |x_2|$ (follows from y and x_2 maximal in $[\![P_2]\!]$). Moreover y is more 'liberal' then x, that is x is an refinement of the partial order in y. We use Proposition 11 and have that $x_2 \in [\![P_2]\!]$ with

$$e_2 <_{x_2} e_2' \iff f(\ell_y(e)) <_{x_1} f(\ell_y(e'))$$

We can then define g on x_2 and have that x_2 and g satisfy Equation 22, Equation 23 and Equation 24.

5. For $x_1 \in [\![P_1]\!]$, let χ, P_2 and $x_2 \in [\![P_2]\!]$ defined above. We have that $[\![P_1|P_2]\!] = \big([\![P_1]\!] \times [\![P_2]\!]\big) \upharpoonright X$ and π_1, π_2 the projections. Denote $[\![P_1|P_2]\!] = (E, C, \ell, \mathsf{P})$ and $[\![P_i]\!] = (E_i, C_i, \ell_i, \mathsf{P}_i)$. Let us show that $\exists z \in [\![P_1|P_2]\!]$ closed with $\pi_1(z) = x_1$, $\pi_2(z) = x_2$ and such that $z \notin X$.

For $e \in E$ let us denote $\pi_1(e) = e_1$ and $\pi_2(e) = e_2$. In the induction hypothesis we also need to show that $\forall e \in z$ the following hold:

$$e_1 \neq \star \tag{26}$$

$$\ell_\chi(e_1) = \ell_z(e) \tag{27}$$

$$e_2 = \star \iff \ell_{x_1}(e_1) = (\alpha, \beta) \tag{28}$$

$$e_2 \neq \star \implies g(e_2) = e_1 \tag{29}$$

From Equation 26 the size of z is equal to the size of x_1. We proceed by induction on the size of the configurations in $[\![P_1]\!]$.

Base case. Let $x_1 = \{e_1\} \in [\![P_1]\!]$.

- If $\ell_{x_1}(e_1) = (\overline{b}\langle a\rangle, c(d))$ then from condition (a) on the precontext we have that there exists $e \in x_1$ such that $e = \mathsf{inst}^s_{x_1}(e_1)$. Hence we cannot have such a configuration $\{e_1\}$ without the instantiator of e_1.
- If $\ell_{x_1}(e_1) = (\overline{b}\langle a\rangle, b(d))$ we have that there exists $e \in E$ such that $\pi_1(e) = e_1$ and $\pi_2(e) = \star$. Straightforward to show that the conditions above hold.
- If $\ell_{x_1}(e_1) = \alpha$ with $\alpha \in \{\overline{b}\langle a\rangle; b(d)\}$ then have that, from condition (b) on χ, $\overline{\alpha} \in \chi$. Moreover from Equation 22 and Equation 16, $\exists\{e_2\} \in [\![P_2]\!]$ such that $\ell_{\{e_2\}}(e_2) = \overline{\alpha}$. It is easy to show that $\{e\} \notin X$, hence $\{e\} \in [\![P_1|P_2]\!]$ closed with $\pi_1(e) = e_1$ and $\pi_2(e) = e_2$. Condition 29 holds from 23 and all the rest hold trivially.

Inductive case. Let $x_1 = x_1' \cup \{e_1\}$ and let z' closed such that $\pi_1(z') = x_1'$. We have the following cases on e_1:

- $\ell_{x_1}(e_1) = (\overline{b}\langle a\rangle, c(d))$. Suppose $b \neq c$ (similar for the other case). Let $e \in E$ with $\pi_1(e) = e_1$ and $\pi_2(e) = \star$. Let $z = z' \cup \{e\}$ and $\forall e' \in z'$, $e' <_z e \iff \pi_1(e') <_{x_1} e_1$. It follows by induction on z' that

$$e' <_z e'' \iff \pi_1(e') <_{x_1} \pi_1(e''). \tag{30}$$

  We show that z is a configuration in $[\![P_1]\!] \times [\![P_2]\!]$, show that it is not in X, that it is closed and then that it satisfies conditions above.

  $z \subseteq E_1 \times E_2$ and z is a partial order (from Equation 30). Also we have that $\pi_1(z) = x_1$. From Equation 30 again and by induction on z' we obtain that $e' <_z e'' \iff \pi_2(e') <_{x_2} \pi_2(e'')$ as $\pi_2(e) = \star$. Hence $\pi_2(z) = \pi_2(z')$.

  From property (a) on the precontext we have that $\ell_\chi(e_1) = (\overline{b}\langle a\rangle, b(d))$. From Equation 27 on z' we obtain that $\forall e' \in z'$, $\ell_\chi(\pi_1(e')) = \ell_z(e)$. Then $\ell_z(e) = (\overline{b}\langle a\rangle, b(d))$ hence z is closed.

  We have to show that $z \notin X$. As z is closed we only need to show that all events in z are compatible according to Definition 15. It follows from conditions (a) and (b) on the precontext χ.

  z trivially satisfies the conditions above.
- $\ell_{x_1}(e_1) = \alpha_1 \neq (\alpha, \beta)$. Using condition (b) on the precontext we have that $\exists\alpha_2 \in \chi$ such that $f(\alpha_2) = e_1$ and $\mathsf{subj}(\alpha_2)\varsigma = \mathsf{subj}(\alpha_1)\varsigma$.

We use then Equation 16 and have that $\alpha_2 \in P_2$, which from Equation 22 implies that $\exists e_2 \in x_2$ such that $\ell_{x_2}(e_2) = \alpha_2$. Also, from Equation 23 we get that

$$g(e_2) = e_1. \tag{31}$$

We have that $e \in E$ with $\pi_1(e) = e_1$, $\pi_2(e) = e_2$. We take $z = z' \cup \{e\}$ with $e' <_z e \iff \pi_1(e') <_{x_1} e_1$. It follows by induction on z' that

$$e' <_z e'' \iff \pi_1(e') <_{x_1} \pi_1(e''). \tag{32}$$

As above, let us show that z a configuration in $[\![P_1]\!] \times [\![P_2]\!]$, that it is not in X, that it is closed and then that it satisfies conditions above.

$z \subseteq E_1 \times E_2$ and from Equation 32 z is a partial order with $\pi_1(z) = x_1$. Also, from Equation 24 $\pi_2(z) = x_2$.
We have that

$$\ell_z(e) = \ell_\chi(e) \tag{33}$$

which follows from Equation 27 on z' and from condition (b) on χ. It implies that $z \notin X$ as well.

Equation 26 and Equation 28 trivially hold. Equation 27 follows from Equation 33, and Equation 29 from Equation 31.

Proof (Theorem 4).

1. $[\![\nu a(\nu b P)]\!] \cong [\![\nu b(\nu a P)]\!]$. We have that $[\![P]\!] \upharpoonright a \upharpoonright b \cong [\![P]\!] \upharpoonright b \upharpoonright a$ where $[\![P]\!] \upharpoonright a \upharpoonright b = [\![P]\!] \upharpoonright X$ and $[\![P]\!] \upharpoonright b \upharpoonright a = [\![P]\!] \upharpoonright X$.
2. $[\![\nu a(P) \mid Q]\!] \cong [\![\nu a(P \mid Q)]\!]$ if $a \notin \mathrm{fn}(Q)$. We show that

$$\mathcal{F}_1 = \big([\![P]\!] \upharpoonright X_1 \times [\![Q]\!]\big) \upharpoonright X_2 \cong \big([\![P]\!] \times [\![Q]\!]\big) \upharpoonright X_3 = \mathcal{F}_2.$$

where $\mathcal{F}_1 = (E_1, C_1, \ell_1, \mathsf{P}_1)$ and $\mathcal{F}_2 = (E_2, C_2, \ell_2, \mathsf{P}_2)$. Let us denote $\pi_{1,P} : \mathcal{F}_1 \to [\![P]\!]$ and $\pi_{1,Q} : \mathcal{F}_1 \to [\![Q]\!]$ and similarly for $\pi_{2,P}$, $\pi_{2,Q}$. Define a bijection on events as follows:

$$\iota(e_1) = e_2 \iff \pi_{1,P}(e_1) = \pi_{2,P}(e_2) \text{ and } \pi_{1,Q}(e_1) = \pi_{2,Q}(e_2).$$

To show it is an isomorphism we show that $x_1 \in \mathcal{F}_1 \iff x_2 \in \mathcal{F}_2$, where $\iota(x_1) = x_2$.

(a) $x_1 \in \mathcal{F}_1 \implies \iota(x_1) \in \mathcal{F}_2$.
We proceed by induction on $<_{x_1}$. Trivial for the base case. For the inductive step, consider $x_1 = x_1' \cup \{e_1\}$ and $\iota(x_1') = x_2'$. We have that $x_1 \notin X_2$ and $x_1 \in [\![P]\!] \upharpoonright X_1 \times [\![Q]\!]$. Hence $\pi_{1,P}(x_1) \notin X_1$. Moreover

$$\pi_{1,P}(x_1) = \pi_{1,P}(x_1') \cup \{\pi_{1,P}(e_1)\} \text{ and } \pi_{1,Q}(x_1) = \pi_{1,Q}(x_1') \cup \{\pi_{1,Q}(e_1)\}.$$

Let $e_2 \in E_2$ such that $\pi_{1,P}(e_1) = \pi_{2,P}(e_2)$ and $\pi_{1,Q}(e_1) = \pi_{2,Q}(e_2)$. Define

$$x_2 = x_2' \cup \{e_2\} \text{ with } e_2' <_{x_2} e_2'' \iff \iota(e_2') <_{x_1} \iota(e_2'').$$

x_2 is then a partial order, with projections defined as x_1 and downward closed in $[\![P]\!]\times[\![Q]\!]$. Hence $x_2 \in [\![P]\!]\times[\![Q]\!]$. Remains to show that $x_2 \notin X_3$, which follows by case analysis from $x_1 \notin X_1$ and from the isomorphism reflecting and preserving the order and the labels.

(b) $x_2 \in \mathcal{F}_2 \implies \iota(x_2) \in \mathcal{F}_1$. We use the same reasoning as above: we proceed by induction to show that $\iota(x_2) \in [\![P]\!] \upharpoonright X_1 \times [\![Q]\!]$. The only difficulty resides in showing that if $\pi_{2,P}(x_2) \in [\![P]\!]$ then $\pi_{2,P}(x_2) \in [\![P]\!] \upharpoonright X_1$. We denote $\pi_{2,P}(x_2)$ with y. We show by contradiction that $\nexists e \in y_p$ such that one of the conditions of Definition 15, Definition 16 holds. Consider the following cases:

i. **[type mismatch]** Suppose $\exists e \in y$ such that $\ell_y(e) = (\alpha, \beta)$ and both α and β are outputs. The argument follow from the fact that the isomorphism preserves the labels: let $e_2 \in x_2$ such that $\pi_{2,P}(e_2) = e$. Then $\pi_{2,P}(\ell_{x_2}(e_2)) = \ell_y(e)$. But as $\ell_y(e)$ is a synchornisation we have that $\ell_{x_2}(e_2) = (\alpha', \beta')$ where α' and β' are both outputs. Then $x_2 \in X_3$ and we reach a contradiction.

ii. **[non unifiable labels]** Let $e \in y$ such that $\ell_y(e) = (\alpha, \beta)$ and $c = \mathsf{subj}(\alpha)$, $b = \mathsf{subj}(\beta)$ with $c \neq b$. Similar to above let $e_2 \in x_2$ such that $\pi_{2,P}(e_2) = e$ and

$$\ell_y(e) = (\alpha, \beta) \implies \ell_{x_2}(e_2) = (\alpha', \beta') \tag{34}$$

We have two cases:

- $\neg\, \mathsf{U}(c, b)$. Then either b or c are in the set of private names of $[\![\nu a(P)]\!]$. To reduce the number of cases, suppose b is not private in $\nu a(P)$ but c is. If $c \neq a$ then we have that $y \notin [\![P]\!]$, contradicting the hypothesis. Remains then to consider the case when $c = a$.
  - $\nexists e' \in y$, $e' = \mathsf{inst}^s_x(e)$. But as π process cannot add an instantiator of b in a parallel composition we have that $b = \mathsf{subj}(\beta')$ and that $\nexists e_2' \in x_2$, $e_2' = \mathsf{inst}^s_y(e_2)$. As $a \in \mathsf{P}_1$ we have that $a \in \mathsf{subj}(\alpha')$. Hence a and b are not unifiable in x_2 which implies that $x_2 \in X_3$. Contradiction.
  - $\nexists e'' \in y$, e'' extruder of e, but $\exists e' \in y$ such that $e' = \mathsf{inst}_y(b)$. Let $e_2' \in x_2$ such that $\pi_{2,P}(e_2') = e'$. From Equation 34 it follows that $a = \mathsf{subj}(\alpha')$. If $x_2 \notin X_3$ then $\mathsf{U}(a, \mathsf{subj}(\beta'))$. We have two possibilities: either $a = \mathsf{subj}(\beta')$ or there exists an extruder of c in x_2.

    In the later case, informally, since a is private in $[\![\nu a.P]\!]$ and $x_2 \in [\![P]\!] \times [\![Q]\!]$ we get that an extruder of a necessarily exists in y.

Let us treat the case $a = \mathsf{subj}(\beta')$ more formally. We have $\ell_{x_2}(e'_2) = (d(b), \overline{d'}\langle a\rangle)$, where $\ell_y(e') = d(b)$. Then there exists an event $e_q \in \pi_{2,Q}(x_2)$ such that its label is $\overline{d'}\langle a\rangle$. Since a is private in $[\![\nu a(P)]\!]$ we have that there exists $e''' \in y$ such that $\ell_y(e''') = \overline{b'}\langle a\rangle$. Let us denote with $e'''_2 \in x_2$ such that $\pi_{2,P}(e'''_2) = e'''$. Then we can show that there exists a chain of synchronisations in x_2 with the least element e'''_2 and the greatest one e_2. As the projection preserves the order we conclude that $e''' <_y e$.

Then e'' is an extruder of e in y and $e'' <_y e$.

- Let us consider the case $c = a$ and let $e' \in y$ such that $b, b' \in \mathsf{subj}(\ell_y(e'))$ for some b' with $\neg\, \mathsf{U}(a, b')$. Consider $e'_2 \in x_2$ such that $\pi_{2,P}(e'_2) = e'$. As $x_2 \notin X_3$, for $b'', b''' \in \mathsf{subj}(\ell_{x_2}(e'_2))$, $\mathsf{U}(a, b'')$ and $\mathsf{U}(a, b''')$. With a reasoning similar to above we get to a contradiction.

  The other cases are similar.

iii. **[disallowed partial event]** Suppose that there exists $e \in y$ such that $\neg\tilde{\tau}_y(e)$ with $a \in \mathsf{subj}(\ell_y(e))$ and $\nexists e' \in x$ extruder of e. We have a similar reasoning as the case above. Since $\exists e_2 \in x_2$ such that $\pi_{2,P}(e_2) = e$, it implies that it necessarily exists an unsynchronised extruder of a in y, smaller then e in y.

We conclude the proof by showing that if $x_2 \in \mathcal{F}_2$ then $x_1 \notin X_1$ similar to the case (a) above.

3. $[\![P \mid Q]\!] \cong [\![Q \mid P]\!]$. We have to show that

$$\big([\![P]\!] \times [\![Q]\!]\big) \restriction X_1 \cong \big([\![Q]\!] \times [\![P]\!]\big) \restriction X_2$$

   which follows from $[\![P]\!] \times [\![Q]\!] \cong [\![Q]\!] \times [\![P]\!]$ and $X_1 = X_2$.

4. $[\![(P \mid Q) \mid R]\!] \cong [\![P \mid (Q \mid R)]\!]$ that is

$$\big(([\![P]\!] \times [\![Q]\!]) \restriction X_1 \times [\![R]\!]\big) \restriction X_2 \cong \big([\![P]\!] \times ([\![Q]\!] \times [\![R]\!]) \restriction X_3\big) \restriction X_4.$$

A.7 Recursion and rigid families

One can deal with recursive definitions on rigid families in a similar manner to their treatment in [22], [9]. Winskel [22] defines a complete partial order on configuration structures and shows that the operations prefix, parallel composition, restriction and sum are continuous with respect to the cpo. A recursive process is then unfolded and each unfolding is encoded in a rigid family. Its denotational semantics is the upper bound of its unfoldings.

Crafa, Varacca, Yoshida [9] define a model for pi calculus where a special treatement for names is necessary. At each unfolding of a process the set of bound names have to be distinct from any other name used in the process, in order to guarantee the Barendregt convention. Moreover the names have to be chosen in an unambiguous manner, hence it is not enough to consider fresh names at each unfolding. Instead the infinite set of bound names used in a process is

fixed in advance and at each unfolding new names are chosen in a consistent way.

Here we revisit the work in [22] and [9] and show that can be adapted to rigid families in a straightforward manner.

Definition 29 (Partial order on rigid families).

$$\begin{aligned}(E_0, C_0) \trianglelefteq (E_1, C_1) \iff & E_0 \subseteq E_1 \\ & C_0 \subseteq C_1 \\ & \forall x \subseteq E_0, x \in C_1 \implies x \in C_0\end{aligned}$$

Lemma 3. *– $\trianglelefteq$ is a partial order on rigid families with least element $(\emptyset, \{\emptyset\})$.*
– A ω chain of rigid families $(E_0, C_0) \trianglelefteq \cdots \trianglelefteq (E_n, C_n) \trianglelefteq \cdots$ has a lub (E, C) defined as

$$\begin{aligned}& E = \cup_{n\in\omega} E_n \\ & x \in C \iff (x \subseteq E \wedge \forall n \in \omega, x_n \in C_n \wedge x = \cup_{n\in\omega} x_n) \\ & \qquad\qquad \text{with } x_n \equiv \cup\{z \in C_n \mid z \subseteq x_n\}\end{aligned}$$

Definition 30 (Monotonic and continuous). *Let* op *a binary operation on rigid families.*

– Let $(E_0, C_0) \trianglelefteq (E_1, C_1)$ and (E, F) rigid families. op *is* monotonic *if*

$$\mathsf{op}\big((E_0, C_0), (E, C)\big) \lhd \mathsf{op}\big((E_1, C_1), (E, C)\big).$$

– Let $(E_0, C_0) \trianglelefteq \cdots \trianglelefteq (E_n, C_n)$ and (E, F) rigid families. op *is* continuous *if*

$$\mathsf{op}\big(\cup_{i\le n}(E_i, C_i), (E, C)\big) = \cup_{i\le n}\mathsf{op}\big((E_i, C_i), (E, C)\big).$$

One can show if an operation is continuous it implies that it is monotonic as well. Hence we prove the following:

Lemma 4. *Prefix, product and restriction are continuous w.r.t $\lhd$.*

CCS We deal with recursive processes by introducing a set of constants, ranged by A, B:

$$P ::= \cdots \mid A$$

and imposing that each constant has a unique definition $A = P_A$. The rule in the LTS of CCS that handles recursion is standard:

$$\frac{P_A \xrightarrow{\alpha} P' \quad A = P_A}{A \xrightarrow{\alpha} P'}$$

The encoding of CCS terms with recursion is now parametric on the level of unfolding:

$$[\![\alpha.P]\!]_k = \alpha.[\![P]\!]_k \quad [\![P|Q]\!]_k = [\![P]\!]_k|[\![Q]\!]_k \quad [\![\nu a(P)]\!]_k = [\![P]\!]_k \upharpoonright X_a \quad [\![A]\!]_k = [\![P_A]\!]_{k+1}$$

The encoding is given in terms of labelled rigid families but it is trivial to extend Definition 29,Definition 30 and Lemma 3,Lemma 4 to the labelled case. We have that $[\![P]\!]_k \lhd [\![P]\!]_{k+1}$, and that $[\![P]\!]_0 = (\emptyset, \emptyset, \emptyset)$ is the bottom element in the ω chain $(\emptyset, \emptyset, \emptyset) \lhd \cdots [\![P]\!]_k \lhd [\![P]\!]_{k+1}$. Hence the interpretation of a process $[\![P]\!]$ is given by $\cup_{n\in\omega}[\![P]\!]_n$.

π calculus Similarly to CCS, we deal with recursive processes by introducing a set of constants, ranged by A, B:

$$P ::= \cdots \mid A(\tilde{a}|\mathbf{z})$$

and imposing that each constant has a unique definition $A(\tilde{a}|\mathbf{z}) = P_A$ where $\tilde{a}$ is the set of distinct names and $\mathbf{z}$ is a mapping to an infinite set of names $\mathbf{z} : \mathbb{N} \to \text{Names}$. $\tilde{a}$ are the free names in P_A and $\mathbf{z}$ are the (infinite) set of bound names in P_A. The set $\mathbf{z}$ is introduced in the semantics of event structures [9] in order to handle the creation of fresh names in the unfolding of A. The set $\tilde{a}$ is specified in order to guarantee distinctness between bound and free names: $\mathbf{z} \cap \tilde{a} = \emptyset$. In this manner we satisfy the Barendregt convention which guarantees the distinctness of bound and free names. We add to the LTS of Figure 11 the following rule:

$$\frac{P_A\{\tilde{b}/\tilde{a}, \mathbf{w}/\mathbf{z}\} \xrightarrow{\alpha} P' \quad A(\tilde{a}|\mathbf{z}) = P_A}{A(\tilde{b}|\mathbf{w}) \xrightarrow{\alpha} P'}$$

with $b \subseteq a$ and $\mathbf{w} \subseteq \mathbf{z}$. Moreover the bound names used at each unfolding need to be unambiguously chosen. As an example consider $A(\tilde{a}|\mathbf{z}) = A_0(\tilde{a_0}|\mathbf{z'}) \mid A_1(\tilde{a_1}|\mathbf{z''})$, with $\tilde{a} = \tilde{a_0} \cup \tilde{a_1}$ and $\mathbf{z'}(n) = \mathbf{z}(2n+1)$, $\mathbf{z''}(n) = \mathbf{z}(2n+2)$.